Vinícius Santino Alves

Um Algoritmo Evolutivo Rápido para Agrupamento de Dados

Dissertação de mestrado apresentada ao Programa de Mestrado em Informática como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação. Área de concentração: Sistemas Inteligentes.

Orientador: Prof. Dr. Ricardo José Gabrielli Barreto Campello
Co-orientador: Prof. Dr. Eduardo Raul Hruschka

Universidade Católica de Santos
Santos, SP
2007

Dados Internacionais de Catalogação
Sistema de Bibliotecas da Universidade Católica de Santos - UNISANTOS
SibiU

A474u Alves, Vinícius Santino
Um Algoritmo Evolutivo Rápido para Agrupamento de Dados / Vinícius Santino Alves - Santos: [s.n.], 2007.
71f, ; 20 cm. (Dissertação de Mestrado - Universidade Católica de Santos, Programa em Informática)

Alves, Vinícius Santino. II. Título.

CDU 681.3:004(043.3)

Vinícius Santino Alves

Um Algoritmo Evolutivo Rápido para Agrupamento de Dados

Dissertação de mestrado apresentada ao Programa de Mestrado em Informática como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação. Área de concentração: Sistemas Inteligentes.

Banca Examinadora:
Prof. Orientador Doutor Ricardo José Gabrielli Barreto Campello
Prof. Doutor André Carlos Ponce de Leon Ferreira de Carvalho
Prof. Doutor Leandro Nunes de Castro Silva

Santos, SP
2007

Resumo

A atividade de agrupamento de dados (obter uma partição que represente a estrutura de um conjunto de objetos) é de vasta aplicabilidade e importância nos dias de hoje. Ferramentas de agrupamento de dados são aplicadas em diversos domínios: inteligência artificial, reconhecimento de padrões, economia, ecologia, psiquiatria, marketing, entre outros.

Algoritmos evolutivos são ferramentas inspiradas na teoria da evolução das espécies que são, em geral, aplicados a problemas de otimização. Tais algoritmos são capazes de encontrar boas soluções (subótimas) em tempo computacional razoável e, por esta razão, eles são utilizados desde a década de 60 como opção para a solução de problemas complexos.

Quando considerado como um problema de otimização combinatória, a atividade de agrupamento de dados tem espaço de busca de complexidade não polinomial. Tal complexidade tem estimulado o desenvolvimento de ferramentas de agrupamento de dados utilizando algoritmos evolutivos. Nesta dissertação apresenta-se o novo Algoritmo Evolutivo Rápido para Agrupamento de Dados (*Fast-EAC*), uma ferramenta capaz de estimar o número ótimo de grupos para um determinado conjunto de dados e a respectiva partição dos dados utilizando a abordagem de algoritmos evolutivos.

Além da proposta do novo *Fast-EAC*, são contribuições desse trabalho a proposta de uma nova metodologia de avaliação para algoritmos evolutivos aplicados a agrupamento de dados e um novo índice externo de avaliação de partições, o *Rand Index* parcial por grupos.

Palavras-chave: Agrupamento de Dados, Algoritmos Evolutivos, k-médias, Estimação automática de número de grupos, *Rand Index* .

Abstract

Data Clustering (finding a data partition in order to represent a structure of a data set) is nowadays a statement of vast applicability and importance. Tools for grouping data are largely used in a great variety of fields, such as artificial intelligence, pattern recognition, economy, ecology, psychiatry, marketing and others.

Evolutionary algorithms are tools inspired by the theory of evolution of species generally used in optimization problems. These algorithms are able to find good solutions within a reasonable computational time and, for this reason, they have been used since the 60s as an alternative to the solution of complex problems.

When considered as a combinatorial optimization problem, clustering has a non-polynomial complexity search space and this complexity has been stimulating the development of tools for clustering using evolutionary algorithms. This dissertation presents the new Fast Evolutionary Algorithm for Clustering (Fast - EAC), a tool able to estimate the number of clusters to a data set and its respective partition using the approach of evolutionary algorithms.

Besides the proposal of the new Fast-EAC, this essay also contributes with a new evaluation methodology to evolutionary algorithms applied to data clustering and a new partition evaluation external index, the partial Rand Index for individual clusters.

Keywords: Data Clustering, Evolutionary Algorithm, k-means, *Rand Index* .

Agradecimentos

Ao meu orientador e co-orientador, Profs. Drs. Ricardo José Gabrielli Campello e Eduardo Raul Hruschka, sou grato pela orientação, trabalho e dedicação; aos colegas e amigos de pós-graduação pelas críticas e sugestões; à minha família e amigos pelo apoio incondicional durante essa jornada; a amiga Jenifer pela revisão; ao LSIn e Fapesp pela infraestrutura que possibilitou esse trabalho; a CAPES, pelo apoio financeiro.

À minha família

Sumário

Lista de Figuras **xv**

Lista de Tabelas **xvii**

Lista de Símbolos **xix**

1 Introdução **1**
1.1 Fundamentos de agrupamento de dados 2
1.1.1 Definição do problema 3
1.1.2 Medidas de similaridade 4
1.1.3 Abordagens clássicas de agrupamento de dados 5
1.2 Fundamentos de algoritmos evolutivos 8
1.3 Organização da dissertação 10

2 Algoritmos evolutivos para agrupamento de dados **13**
2.1 Algoritmos evolutivos aplicados ao problema de agrupamento para número de grupos k fixo 13
2.2 Algoritmos evolutivos aplicados ao problema de agrupamento de dados com estimação automática do número de grupos k 16
2.3 Resumo 18

3 Algoritmo evolutivo rápido para agrupamento de dados *Fast- EAC* **19**
3.1 Algoritmo *EAC* 19
3.1.1 Operadores de mutação 21
3.1.2 Busca local pelo algoritmo das k-médias 25
3.1.3 População inicial, seleção e passos principais 26
3.2 Melhorias propostas para o *EAC* 27
3.2.1 *EAC*-I 27
3.2.2 *EAC*-II 32
3.2.3 *EAC*-III 32
3.2.4 *EAC*-IV 34
3.2.5 *EAC*-V 36
3.2.6 Análise de eficiência computacional dos algoritmos propostos 37
3.3 Funções de adequabilidade 37

3.3.1 *Rand Index* . . . 38
3.3.2 Silhueta Simplificada . . . 39
3.4 Algoritmo *Fast-EAC* . . . 40
3.5 Resumo . . . 41

4 Experimentos **43**
4.1 Avaliação experimental . . . 43
4.1.1 Conjunto de dados . . . 43
4.1.2 Metodologia . . . 44
4.1.3 Resultados e análise experimental . . . 46
4.2 Experimento adicional . . . 49
4.2.1 Metodologia . . . 49
4.2.2 Experimentos e análise de resultados . . . 52
4.3 Resumo . . . 54

5 Conclusões **55**

Referências bibliográficas **57**

A Propriedades do *Rand Index* para grupos individuais **63**

Lista de Figuras

1.1 Dois conjuntos: (a) conjunto de dados com grupos bem definidos; (b) conjunto de dados com grupos sobrepostos. 7
1.2 Dendrograma resultante de um algoritmo de agrupamento hierárquico. 8

3.1 Esquema de codificação do *EAC*. 20
3.2 6 genótipos representando a mesma partição de dados. 21
3.3 Exemplo de resultado indesejável de operadores de recombinação padrão (insensibilidade ao contexto). 22
3.4 Exemplo de aplicação do operador de mutação OM_1: (a) Partição original codificada pelo genótipo; (b) É selecionado um grupo para mutação, sendo este representado pelo rótulo '2'; (c) Genótipo resultante da aplicação do operador de mutação OM_1. . 22
3.5 Exemplo de aplicação do operador de mutação OM_2: (a) Partição original codificada pelo genótipo; (b) É selecionado um grupo para mutação, sendo este representado pelo rótulo '1'; (c) Indica-se quais objetos do grupo '1' são mais próximos de quais sementes; (d) Genótipo resultante da aplicação do operador de mutação OM_2. 24
3.6 Duas partições. (a) partição adequada à busca guiada; (b) partição não adequada à busca guiada. 28
3.7 Avaliação de funções de probabilidade para seleção de grupos: (a) Função complemento da probabilidade de não mutar, primeiro cenário; (b) Função complemento da probabilidade de não mutar, segundo cenário; (c) Função inverso da função de adequabilidade, primeiro cenário; (d) Função inverso da função de adequabilidade, segundo cenário. 31
3.8 Normalização linear dos valores de adequabilidade parcial: (a) Probabilidade de mutar para o primeiro cenário; (b) Probabilidade de mutar para o segundo cenário; . . . 32
3.9 (a) Base de dados; (b) Valores de função de adequabilidade para partições com diferentes números de grupos, k. 33
3.10 Genótipo $\mathbf{G}_1$. 35
3.11 Dois conjuntos de dados: (a) exemplo em que é adequado o *EAC*-IV; (b) exemplo onde o *EAC*-IV mostra-se inadequado. 36

4.1 Conjunto 9-Gauss. 48

Lista de Tabelas

1.1 Medidas de dissimilaridade para dados contínuos. 5

3.1 Estrutura auxiliar baseada em centróides utilizada pelo *EAC*-IV. 35

4.1 Valores de referência para avaliação de partições (k_{RI} e k_{SS} indicam valores típicos de número de grupos encontrados pelo *Rand Index* e Silhueta Simplificada, respectivamente). 45
4.2 Tempos Computacionais (*Rand Index* - 4 Genótipos). 48
4.3 Tempos Computacionais (*Rand Index* - 20 Genótipos). 48
4.4 Tempos Computacionais (Silhueta Simplificada - 4 Genótipos). 49
4.5 Tempos Computacionais (Silhueta Simplificada - 20 Genótipos). 50
4.6 Resultados da busca seqüencial: média (μ) e desvio padrão (σ) dos tempos computacionais. 53
4.7 Sumário de tempo de processamento (segundos): média (μ) e desvio padrão (σ). . . . 53

Lista de Símbolos

$\mathbf{X}$	-	Conjunto de dados
N	-	Número de objetos do conjunto de dados (Cardinalidade)
k	-	Número de grupos da partição
$\mathbf{C}_i$	-	i-ésimo *Cluster* ou Grupo
T	-	Número de genótipos da população
$\mathbf{G}_i$	-	Genótipo de Índice i
$\mathbf{x}_j$	-	Objeto j do conjunto de dados
$s(\mathbf{x}_i)$	-	Silhueta do objeto $\mathbf{x}_i$
$\Omega(\mathbf{R}, \mathbf{C})$	-	Rand Index das partições $\mathbf{R}$ e $\mathbf{C}$
$\overline{P}$	-	Complemento da função de probabilidade P

Capítulo 1

Introdução

A tarefa de agrupamento de dados envolve a organização de um conjunto de objetos em grupos (categorias ou *clusters*) formados com base em suas similaridades [Jain & Dubes, 1988]. O objetivo da atividade de agrupamento de dados é formar grupos caracterizados por alta homogeneidade intra-grupo e heterogeneidade inter-grupos, ou seja, cada objeto deve ser parecido com os objetos do grupo ao qual pertence e os objetos pertencentes a grupos distintos devem ser mais diferentes quanto possível.

A classificação de objetos similares em grupos é uma importante atividade humana. De forma corriqueira é possível se deparar com agrupamentos naturais comuns à sociedade: grupos de pessoas conhecidas e desconhecidas, grupos de pessoas do sexo masculino e sexo feminino, etc. Nas ciências, a categorização em grupos sempre desempenhou papel essencial. No século XVIII, Linnaeus e Sauvages classificaram de forma extensiva animais, plantas, minerais e doenças [Kaufman & Rousseeuw, 1990]. Nos dias de hoje, diversas aplicações podem ser encontradas, como por exemplo em marketing. Em [Reinschmidt et al., 1999] é sugerida a utilização de ferramentas de agrupamento de dados para segmentação de clientes e detecção de fraudes. Aplicações industriais de agrupamento de dados podem ser encontradas em [Falkenauer, 1998]. Outras aplicações de agrupamento de dados abrangem áreas como astronomia, mineração de textos, mineração de dados, etc (para um visão geral de aplicações de agrupamento de dados veja [Jain & Dubes, 1988, Everitt et al., 2001]).

A existência de aplicações bem sucedidas de algoritmos inspirados na Teoria da Evolução das Espécies (algoritmos evolutivos) [Bäck et al., 2000, Davis, 1996, Mitchell, 1996] em atividades como planejamento, simulação, identificação, classificação e controle, aliada à grande complexidade computacional inerente à tarefa de agrupamento de dados, tem motivado o desenvolvimento e aprimoramento de versões dedicadas desses algoritmos para tratar especificamente o problema de agrupamento. As primeiras aplicações nesse sentido datam do início da década de 90 e desde en-

tão tem-se obtido resultados substancialmente relevantes [Falkenauer, 1998, Hruschka et al., 2004b, Naldi, 2006].

Nas seções seguintes deste capítulo serão apresentados os aspectos fundamentais que tangem a tarefa de agrupamento de dados e as ferramentas de algoritmos evolutivos, de forma independente.

1.1 Fundamentos de agrupamento de dados

Diversas técnicas já foram desenvolvidas para tratar o problema de agrupamento de dados apoiando-se em abordagens distintas, cada uma delas sendo mais adequada a um conjunto específico de problemas. Uma revisão sobre os principais algoritmos existentes e as diversas formas de atacar o problema é encontrada em [Jain et al., 1999, Xu & Wunsch, 2005]. Baseado nestes autores, é possível classificar uma ferramenta para agrupamento de dados de forma resumida como:

1. Particional ou hierárquica: ferramentas particionais fornecem uma única partição dos dados em função de um número específico k de grupos, definido ou não *a priori*. Um método hierárquico, por sua vez, fornece um conjunto de partições onde cada partição subseqüente é descendente da anterior, formando uma estrutura hierárquica aninhada de agrupamentos particionais.

2. Monotética ou politética: ferramentas monotéticas utilizam um atributo por vez para o agrupamento, enquanto ferramentas politéticas utilizam conjuntamente todos os atributos no processo de agrupamento.

3. Determinística ou estocástica: ferramentas determinísticas são procedimentos fechados, em que cada estado do processo de busca pode ser determinado pelo estado anterior, enquanto ferramentas estocásticas dependem de algum tipo de aleatoriedade, de forma que um estado não pode ser definido observando apenas o estado anterior.

4. Incremental ou não incremental: ferramentas incrementais podem trabalhar com subconjuntos do conjunto de dados, enquanto ferramentas não incrementais agrupam todo o conjunto de uma vez. Ferramentas incrementais são de particular interesse para aplicações de mineração de dados em que o conjunto de dados é geralmente muito grande e se faz necessário o agrupamento por amostras dos dados.

5. Grupos disjuntos ou sobrepostos: ferramentas de agrupamento de dados que buscam por estruturas formadas por grupos disjuntos de dados são caracterizadas pela restrição de que um objeto pertença exatamente a um grupo. Por sua vez, ferramentas que buscam por estruturas formadas por grupos sobrepostos distinguem-se por contemplar, em seu espaço de estados, partições com objetos pertinentes a múltiplos grupos.

No presente trabalho o problema de agrupamento de dados é abordado sob o paradigma de métodos particionais, politéticos, estocásticos, não incrementais e que buscam por partições de dados em subconjuntos disjuntos. Mais especificamente, a atividade de agrupamento de dados será tratada como um problema de otimização. Considerando uma função f, capaz de avaliar a qualidade de uma dada partição de dados, sua otimização resultará num melhoramento da qualidade da partição.

1.1.1 Definição do problema

Formalmente, o problema de agrupamento de dados pode ser definido, sob o ponto de vista particional, como a obtenção de k subconjuntos (grupos ou categorias) de um conjunto de dados $\mathbf{X}$ com N objetos $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$, em que cada $\mathbf{x}_i \in \Re^\rho$ é um vetor ρ-dimensional. Os objetos $\mathbf{x}_i$ $i = \{1, 2, ..., N\}$ deverão ser agrupados em k categorias não vazias e disjuntas $\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_k\}$ tal que:

$\mathbf{C}_1 \cup \mathbf{C}_2 \cup \ldots \cup \mathbf{C}_k = \mathbf{X}$; $\quad \mathbf{C}_j \neq \emptyset$; $\quad \mathbf{C}_i \cap \mathbf{C}_j = \emptyset \quad i \neq j$

É esperado que a partição $\mathbf{C}$ obtida represente o conjunto de dados $\mathbf{X}$ de forma tal que os objetos pertencentes a dado grupo $\mathbf{C}_i$ sejam mais similares entre si que dos objetos pertencentes aos demais grupos $\mathbf{C}_j$, $i \neq j$. A mensuração da similaridade entre os objetos é feita de forma indireta por meio do cálculo das distâncias segundo alguma medida adequada.

É possível calcular o número de partições distintas possíveis em um determinado conjunto de dados $\mathbf{X}$ com N objetos, quando se deve particioná-lo em k grupos, por meio da seguinte expressão [Jain & Dubes, 1988]:

$$S(N,k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}\binom{k}{i}(i)^N \tag{1.1}$$

A equação (1.1) sugere que o problema de agrupamento de dados tem espaço de busca de tamanho não polinomial em relação ao tamanho da entrada (número N de objetos e de grupos k). Ao considerar que a otimização das partições envolve a estimação automática de k, o espaço de busca é ampliado para $\sum_{i=1}^{N} S(N,i)$, denotando que, mesmo para instâncias pequenas do problema, soluções baseadas na enumeração e na avaliação a partir de uma função objetivo de todas as partições possíveis torna-se computacionalmente inviável. Considerando por exemplo uma instância do problema com $N = 19$ e $k = 4$, obtém-se aproximadamente 11×10^9 partições possíveis. Essa dificuldade tem estimulado a pesquisa por algoritmos eficientes de aproximação, incluindo não somente o uso de heurísticas *ad hoc* para problemas específicos como também a utilização de metaheurísticas [Rayward-Smith, 2005] com aplicabilidade mais abrangente. Particularmente, algoritmos evolutivos são metaheurísticas amplamente aceitas como efetivas para tratar problemas NP-difíceis, sendo capazes de prover boas soluções (sub-ótimas) em um tempo computacional razoável [Eiben & Smith, 2003]. Sob essa pers-

pectiva, algoritmos evolutivos para agrupamento de dados têm sido propostos na literatura – veja [Falkenauer, 1998] para um visão geral.

1.1.2 Medidas de similaridade

Muito do esforço para produzir uma estrutura mais simples que represente um conjunto de dados complexo requer uma medida de "proximidade" ou "similaridade". Em geral, existe uma grande subjetividade envolvida na escolha da medida de similaridade. Considerações importantes incluindo a natureza das variáveis (discretas, continuas, binárias), escalas das medidas (nominal, ordinal, intervalar) e conhecimento de domínio são, entre outras, informações relevantes na escolha de uma medida de similaridade adequada. Normalmente, na tarefa de agrupamento de dados, a similaridade entre dois objetos é calculada de forma indireta por meio de uma medida de distância.

A medida de distância Euclidiana é uma das medidas mais utilizadas para a atividade de agrupamento de dados [Everitt et al., 2001]. Sua larga difusão está intimamente ligada ao baixo custo computacional a ela associado, $O(\rho)$, e à sua simplicidade de interpretação (i.e, a distância física entre dois objetos ρ-dimensionais no espaço Euclidiano). Porém, essa medida de distância pode ser inadequada dependendo do contexto, uma vez que cada atributo contribui igualmente para o cálculo [Johnson & Wichern, 1998] entre outros. Quando os atributos representam medidas de diferentes magnitudes considerá-los de forma igual implicará em que os atributos de maior magnitude terão mais influência sobre o agrupamento do que os demais. Uma estratégia simples para evitar possíveis problemas com a magnitude dos atributos consiste em modificar a escala (normalizar) de forma que todos os atributos sejam definidos dentro do mesmo intervalo [Everitt et al., 2001]. No entanto, deve-se ter em mente que se diferenças nas escalas não forem inerentes às respectivas grandezas, mas decorrentes de problemas amostrais na base de dados, o procedimento de normalização pode inserir distorções.

Além da medida de distância Euclidiana, que será adotada neste trabalho (utilizando a normalização dos atributos) pelas razões já discutidas, diversas outras medidas podem ser utilizadas para tentar representar a estrutura dos dados mais adequadamente. Na Tabela 1.1 são apresentadas algumas das principais medidas de dissimilaridade para atributos contínuos.

De maneira sintética, as medidas apresentadas na tabela 1.1 podem ser classificadas em dois grandes grupos: o primeiro correspondente às medidas de distância propriamente ditas e o segundo grupo correspondente às medidas de correlação.

A medida Euclidiana é a mais utilizada dentre as medidas baseadas em distância. Tanto a medida Euclidiana ($r = 2$) quanto a medida de Manhattan ($r = 1$) correspondem a casos particulares da medida de Minkowski, sendo a medida Euclidiana a distância direta entre dois objetos (hipotenusa do triângulo formado quando $\rho = 2$) e a distância de Manhattan a soma das diferenças para cada um dos

atributos (soma dos catetos do triângulo formado quando $\rho = 2$). A medida de distância de Canberra, por sua vez, tem sua maior utilidade ligada à sua capacidade de tratamento "natural" para objetos que possuam atributos contínuos e binários [Everitt et al., 2001].

As medidas de correlação, como a correlação de Pearson e a separação angular (ou Medida do Cosseno), são medidas de dissimilaridade derivadas de coeficientes de correlação. No caso da medida de separação angular, a distância entre dois objetos é baseada no cosseno do ângulo entre os dois vetores da origem até os objetos. A medida de correlação de Pearson possui interpretação similar, diferindo da anterior porque os vetores têm sua origem na média das ρ dimensões dos objetos.

A adoção de uma medida baseada em distâncias ou em um coeficiente de correlação está intimamente ligada à natureza dos dados. Por exemplo, a utilização de medidas baseadas em distância são mais adequadas quando os atributos estão em diferentes magnitudes e quando desprezar a magnitude entre os objetos implica em ignorar informação relevante.

Tabela 1.1: Medidas de dissimilaridade para dados contínuos.

Medida	Fórmula
Distância Euclidiana	$d_{ij} = (\sum_{l=1}^{\rho}(\mathbf{x}_{il} - \mathbf{x}_{jl})^2)^{\frac{1}{2}}$
Distancia de Manhattan	$d_{ij} = \sum_{l=1}^{\rho} \lvert\mathbf{x}_{il} - \mathbf{x}_{jl}\rvert$
Distância de Minkowski	$d_{ij} = (\sum_{l=1}^{\rho} \lvert\mathbf{x}_{il} - \mathbf{x}_{jl}\rvert^r)^{\frac{1}{r}}\ (r \geq 1)$
Distância de Canberra	$d_{ij} = \begin{cases} 0 & \forall \mathbf{x}_{il}, \mathbf{x}_{jl} \mid \mathbf{x}_{il} = \mathbf{x}_{jl} = 0 \\ \sum_{l=1}^{\rho} \frac{\lvert\mathbf{x}_{il} - \mathbf{x}_{jl}\rvert}{\lvert\mathbf{x}_{il}\rvert + \lvert\mathbf{x}_{jl}\rvert} & \forall \mathbf{x}_{il}, \mathbf{x}_{jl} \mid \mathbf{x}_{il} \neq 0 \vee \mathbf{x}_{jl} \neq 0 \end{cases}$
Correlação de Person	$d_{ij} = (1 - \phi_{ij})/2$ $\phi_{ij} = \sum_{l=1}^{\rho}(\mathbf{x}_{il} - \bar{x}_{i.})(\mathbf{x}_{jl} - \bar{x}_{j.}) / (\sum_{l=1}^{\rho}(\mathbf{x}_{il} - \bar{x}_{i.}) \sum_{l=1}^{\rho}(\mathbf{x}_{jl} - \bar{x}_{j.}))^{\frac{1}{2}}$ $\bar{x}_{i.} = \frac{1}{\rho}\sum_{l=1}^{\rho}\mathbf{x}_{il}$
Separação Angular	$d_{ij} = (1 - \phi_{ij})/2$ $\phi_{ij} = \sum_{l=1}^{\rho}\mathbf{x}_{il}\mathbf{x}_{jl} / (\sum_{l=1}^{\rho}\mathbf{x}_{il} \sum_{l=1}^{\rho}\mathbf{x}_{jl})^{\frac{1}{2}}$

1.1.3 Abordagens clássicas de agrupamento de dados

A seguir apresenta-se uma visão geral das principais abordagens para agrupamento de dados.

Agrupamento Particional

Métodos de agrupamento de dados particionais buscam a construção de k grupos, com as seguintes restrições:

- Cada grupo contém ao menos um objeto.
- Cada objeto pertence exatamente a um grupo (grupos disjuntos).

Formalmente, $u_{ij} \in \{0,1\}\, \forall i,j$, $\sum_{i=1}^{k} u_{ij} = 1\ \forall j$ e $\sum_{j=1}^{N} u_{ij} > 0\ \forall i$, onde u_{ij} é a pertinência do j-ésimo objeto ao i-ésimo grupo.

Essas restrições implicam que o número máximo de grupos k é o número de objetos ($k \leq N$). Como o número de grupos está intimamente ligado à estrutura dos dados, ou seja, nem todos os valores de k levam a partições "naturais" dos dados, uma estratégia baseada em obter múltiplas partições para diversos valores de k e avaliá-las, seja por um especialista ou por um critério numérico, é uma das possíveis abordagens quando o número k de grupos não é conhecido *a priori*. Nesse caso, reduz-se o problema de agrupamento de dados a um problema de otimização, no qual o resultado esperado é o "melhor" dentre um conjunto de partições, a depender do critério numérico utilizado.

Agrupamento com sobreposição (*overlapping*)

Em muitos casos o conjunto de dados não tem sua estrutura bem definida, o que impossibilita representá-la adequadamente com uma partição de k grupos disjuntos. Nestes casos, relaxar a restrição de que cada objeto pertence exatamente a um grupo pode ser mais adequado, permitindo representar o conjunto de forma mais coerente com sua estrutura intrínseca. Na Figura 1.1 são ilustrados casos em que se faz adequado representar a estrutura do conjunto de dados utilizando partições com grupos disjuntos e partições com grupos sobrepostos (*overlapping*). Na Figura 1.1(a) é apresentado um conjunto de dados com quatro grupos de distribuição Gaussiana, facilmente detectáveis por inspeção visual, os quais são bastante separados entre si. Na Figura 1.1(b) é apresentado um dos conjuntos de dados apresentado em [Campello & Hruschka, 2006], em que é possível reconhecer, com inspeção visual, nove grupos oriundos de distribuição Gaussiana bidimensional. Nesse caso, porém, percebe-se que a fronteira entre os grupos não é bem delimitada. De fato, a representação gráfica sugere que para alguns objetos não é possível discriminar qual o seu grupo mais pertinente.

Uma abordagem bastante difundida para representar partições de dados com grupos sobrepostos, como aqueles apresentados na Figura 1.1(b), é a utilização de partições *fuzzy* (nebulosas) dos dados. Nesta abordagem, considera-se que cada objeto pertence a todos os conjuntos com alguma pertinência (possivelmente nula). Formalmente, partições *fuzzy* dos dados usualmente obedecem às seguintes restrições [Bobrowski & Bezdek, 1991]:

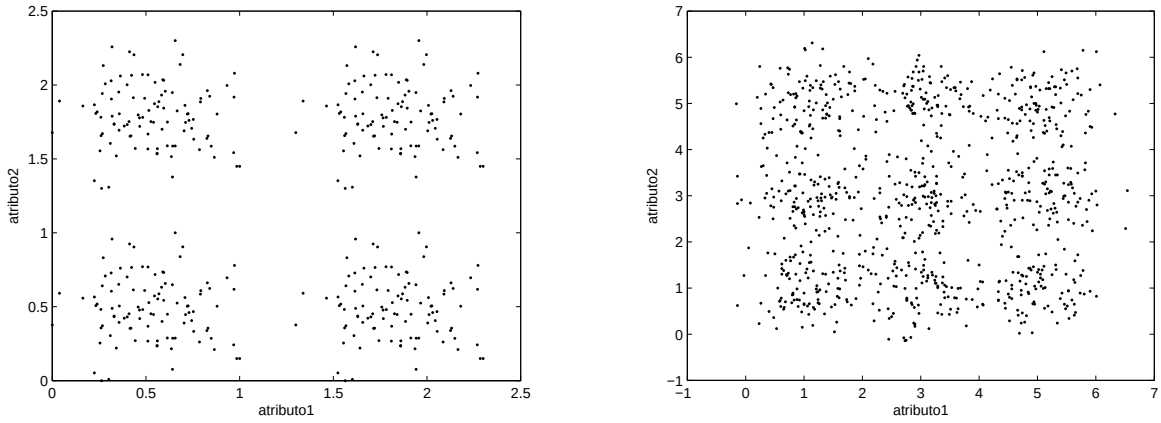
$$\begin{aligned} &0 \leq u_{ij} \leq 1\ \ \forall i,j \\ &\textstyle\sum_{i=1}^{k} u_{ij} = 1\ \ \forall j \\ &0 < \textstyle\sum_{j=1}^{N} u_{ij} < N\ \ \forall i \end{aligned} \tag{1.2}$$

onde $\mu_{ij} \in [0,1]$, sendo:

- u_{ij}: pertinência do j-ésimo objeto do conjunto ao i-ésimo grupo *fuzzy* (*fuzzy cluster*).

- N: número de objetos do conjunto.
- k: número de grupos.

Por fim, como a estrutura dos dados não é conhecida *a priori* e a escolha de uma abordagem adequada para a representação do conjunto de dados é relacionada a esta estrutura, como exemplificado anteriormente, até o que se sabe não é possível determinar antecipadamente qual deverá ser a abordagem adotada (particional ou *overlapping*) em um caso prático quando não se dispõe de qualquer informação da natureza ou estrutura do conjunto de dados.



(a) (b)

Figura 1.1: Dois conjuntos: (a) conjunto de dados com grupos bem definidos; (b) conjunto de dados com grupos sobrepostos.

Agrupamento Hierárquico

Técnicas de agrupamento hierárquico de dados apresentam-se tanto como uma série de sucessivas aglomerações quanto como uma série de sucessivas divisões dos objetos em grupos. Métodos hierárquicos aglomerativos inicialmente dispõem cada objeto como um grupo por si só (*singleton*) realizando então sucessivas aglomerações guiadas por uma medida de dissimilaridade (gerando uma hierarquia de partições) até resultar em um grupo que contém todos os objetos. Métodos divisivos partem de uma partição com um único grupo. A partir deste são geradas sucessivas divisões com base em uma medida de dissimilaridade, resultando em uma hierarquia de partições. Na Figura 1.2 é ilustrado um agrupamento hierárquico para uma instância de $N = 30$ objetos. Na base do dendrograma apresentam-se os trinta *singletons* e, nos níveis superiores, a estrutura do conjunto cada vez mais condensada, até resultar em uma partição com um único grupo. Observe na Figura 1.2 que, ao considerar individualmente uma partição resultante da secção na estrutura hierárquica com distância

(no eixo vertical) de 0,8, por exemplo, obtém-se uma partição dos dados com dois conjuntos: o primeiro formado pelos objetos {20, 25, 24, 9, 27, 2, 17, 21, 4, 22, 16, 7, 28, 11, 8, 26, 30} e o seguinte formado pelos objetos {1, 5, 12, 3, 14, 13, 23, 10, 15, 6, 19, 18, 29}.

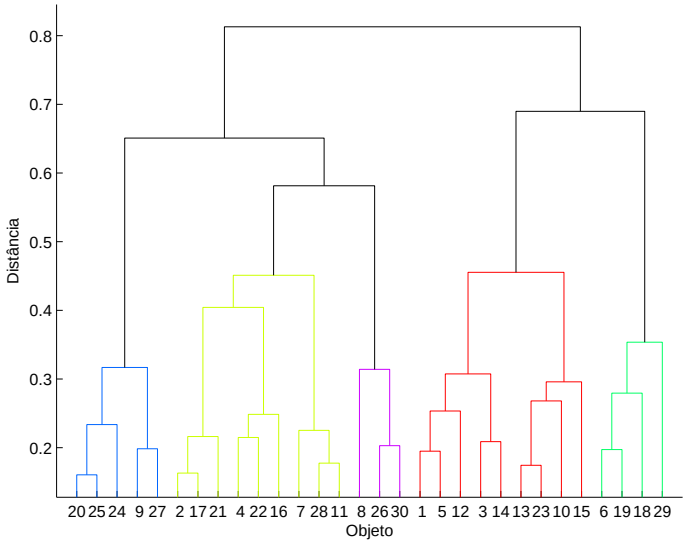


Figura 1.2: Dendrograma resultante de um algoritmo de agrupamento hierárquico.

Algoritmos hierárquicos de agrupamento de dados são de larga aplicação [Jain & Dubes, 1988, Johnson & Wichern, 1998], no entanto tais ferramentas tornam-se proibitivas para grandes instâncias de problemas, tais como aplicações de mineração de grandes bases de dados, onde prefere-se a utilização de métodos particionais por esses serem em geral de menor complexidade de tempo e espaço em comparação aos métodos hierárquicos [Jain et al., 1999].

1.2 Fundamentos de algoritmos evolutivos

A teoria da Evolução foi proposta por Charles Darwin e Alfred Russel Wallace. Em 1859 Darwin publicou “A Origem das Espécies”, em que apresenta a teoria da seleção natural [Lopes, 2003], na qual os indivíduos melhor adaptados ao meio ambiente vencem a luta pela sobrevivência e conseguem deixar mais descendentes. Darwin, entretanto, não conseguiu distingüir as características hereditárias das não hereditárias. Outro trabalho contemporâneo ao de Darwin, o do monge Gregor Johamn Mendel, esclareceu o mecanismo pelo qual os caracteres são transmitidos entre as gerações [Lopes, 2003]. Em seu trabalho com ervilhas, Mendel, inferiu que as características eram compostas por dois fatores e que um indivíduo possuía um fator materno e outro paterno. Hoje esses fatores são denominados como genes. A união das teorias de Mendel sobre genética e a evolução de Darwin originaram a

teoria convencionada como neo-darwinismo. É importante esclarecer que toda variação observada por Darwin nas espécies não se deve unicamente aos fatores genéticos, o genótipo do indivíduo, mas também ao ambiente. Sendo assim, o fenótipo, que representa as variações visíveis, é a junção do genótipo e da variação ambiental. A computação evolutiva é a área da computação representada pelos algoritmos inspirados no neodarwinismo. A idéia fundamental da computação evolutiva é aproveitar a vocação natural desta teoria para otimização (de populações de indivíduos) em aplicações à problemas computacionais.

Especificamente em algoritmos evolutivos, uma população representa um conjunto finito de tamanho T de soluções candidatas a determinado problema, sendo cada *indivíduo*, ou *genótipo* por simplicidade, uma solução codificada utilizando determinado alfabeto. Tal como os indivíduos na teoria da Evolução, cada solução candidata compete com as demais em um ambiente pela sua perpetuação. Em algoritmos evolutivos uma função objetivo, chamada função de adequabilidade ou adaptabilidade (*fitness*), desempenha o papel do ambiente. Quanto melhor a avaliação de um indivíduo pela função de adequabilidade, mais adaptado ele é ao ambiente e possui maior probabilidade de sobreviver e gerar mais descendentes.

Trabalhos utilizando algoritmos evolutivos datam da década de 60. Em [Holland, 1969] foram propostos os algoritmos genéticos como um mecanismo de busca eficiente. Em [Rechenberg, 1973] e [Schwefel, 1977] foi proposta a utilização de estratégias evolutivas para problemas de otimização envolvendo parâmetros reais. No trabalho [Fogel et al., 1966] foi criado o conceito de programação evolutiva, que junto com as anteriormente citadas, formam a base principal dos chamados algoritmos evolutivos. Uma visão histórica sobre algoritmos evolutivos pode ser encontrada em [Mitchell, 1996, Fogel, 1999, Bäck et al., 1997]

Embora existam diversas abordagens classificadas como algoritmos evolutivos, é possível de uma maneira geral apresentar a estrutura básica que cerne este tipo de ferramenta: dada uma população de indivíduos, a pressão seletiva realiza a seleção natural (sobrevivência dos mais aptos), que por sua vez causa o crescimento da adaptação geral da população. Em termos computacionais, dada uma função de adequabilidade a ser otimizada e uma população inicial que pode ser gerada aleatoriamente, avalia-se a população inicial pela função de adequabilidade. Com base na adequabilidade dos indivíduos da população, são selecionados os indivíduos que serão fonte para a próxima geração. Um operador de recombinação (reprodução sexuada ou *crossover* do inglês) pode ser aplicado a dois ou mais indivíduos resultando em um ou mais filhos. Um operador de mutação pode ser aplicado sobre um indivíduo resultando em um novo indivíduo com características genotípicas (e usualmente fenotípicas) distintas daquelas do indivíduo original. Tais novos indivíduos, resultantes do processo de recombinação e/ou mutação, irão competir com seus pais para compor a próxima geração. No Algoritmo 1 apresenta-se um esquema sintético de um algoritmo evolutivo.

Algoritmo 1 Pseudo-código de algoritmo evolutivo genérico.

```
1: [Melhor Genótipo] = Algoritmo Evolutivo();
2: Inicie a população com genótipos aleatórios;
3: Avalie cada genótipo (solução candidata) segundo uma função de adequabilidade;
4: Enquanto Condição de parada não é atingida faça
5:     Selecione os pais;
6:     Aplique os operadores evolutivos apropriados (recombinação e/ou mutação);
7:     Avalie os novos genótipos;
8:     Selecione os indivíduos para a próxima geração;
9: Fim enquanto
```

Pode-se considerar que, nos algoritmos evolutivos, a escolha do método de seleção, do operador de recombinação e do operador de mutação, é uma questão de projeto. Em [Bäck et al., 2000, Fogel, 1999, Davis, 1996, Mitchell, 1996] encontram-se diversos exemplos destes operadores e suas principais aplicações. A função de adequabilidade, por sua vez, é uma questão de projeto inerente ao problema que se deseja otimizar.

Por sua natureza não determinística (estocástica), esses algoritmos são de particular interesse em aplicações em problemas de grande complexidade, uma vez que é bem conhecido que algoritmos evolutivos são capazes de prover boas soluções aproximadas em tempo factível [Mitchell, 1996].

1.3 Organização da dissertação

A dissertação está organizada da seguinte forma. Nesse capítulo introdutório foram apresentados os conceitos fundamentais de agrupamento de dados e algoritmos evolutivos. Tais conceitos fornecem a base necessária para os pontos específicos abordados neste trabalho.

No Capítulo 2 serão apresentados alguns dos principais algoritmos para agrupamento de dados particional que utilizam o paradigma de computação evolutiva. Este capítulo será seccionado em dois tópicos principais: os algoritmos que necessitam da definição prévia do número de grupos na partição resultante do agrupamento; os algoritmos que são capazes de estimar o número k de grupos mais adequado para representar a estrutura do conjunto, enquanto otimizam a partição dos dados. Tal capítulo possibilitará uma visão geral do estado da arte destas classes de algoritmos.

Um conjunto de cinco novos algoritmos será apresentado no Capítulo 3. A proposta desses algoritmos é uma iniciativa com o objetivo de melhorar o desempenho do algoritmo *EAC* em termos de tempo de processamento. Como síntese dos novos algoritmos que apresentam melhores resultados nas análises experimentais em relação ao *EAC*, projeta-se um novo algoritmo evolutivo para agrupamento de dados denominado *Fast-EAC* . Esse algoritmo constitui o principal produto dessa dissertação. Em termos gerais, todos os demais capítulos aqui abordados fornecem base, contextuali-

zação ou ferramentas de avaliação para os algoritmos propostos no Capítulo 3.

O Capítulo 4 abordará a avaliação empírica dos algoritmos desenvolvidos no Capítulo 3. São constituintes desse capítulo a descrição da metodologia de avaliação, uma breve descrição dos conjuntos de dados, o sumário dos resultados dos experimentos e suas respectivas análises.

Conclusões sobre os algoritmos e metodologias propostos neste trabalho constam no Capítulo 5, que inclui as perspectivas de pesquisas decorrentes deste trabalho.

Capítulo 2

Algoritmos evolutivos para agrupamento de dados

Com o intuito de propiciar uma visão geral sobre ferramentas de algoritmos evolutivos aplicadas a agrupamento de dados, serão apresentadas algumas das iniciativas de pesquisa mais relevantes até o momento nesse contexto.

Diversas contribuições científicas já foram realizadas utilizando algoritmos evolutivos aplicados ao problema de agrupamento de dados. De forma geral, essas abordagens podem ser classificadas como algoritmos com número k de grupos fixo e algoritmos com determinação automática do número k de grupos. Apesar de muitos trabalhos relevantes terem sido direcionados para solução do problema de agrupamento de dados, uma das atividades mais críticas, a estimação do número correto de grupos, ainda não foi devidamente explorada. A literatura é em certo grau carente de bons algoritmos capazes de resolver o problema de fornecer em um tempo razoável uma partição que represente a estrutura dos dados quando não é conhecido o número correto de grupos. Dentre alguns trabalhos mais relevantes direcionados à solução deste problema pode-se destacar [Bandyopadhyay & Maulik, 2002, Liu et al., 2004, Casillas et al., 2003, Hruschka & Ebecken, 2003, Hruschka et al., 2004b, Hruschka et al., 2004c, Hruschka et al., 2006].

2.1 Algoritmos evolutivos aplicados ao problema de agrupamento para número de grupos k fixo

Em [Lucasius et al., 1993] foi proposto um algoritmo genético para solução do problema de agrupamento de dados em k grupos utilizando objetos do conjunto de dados como representantes dos grupos (medóides), codificando cada genótipo como um vetor de k (arbitrado) posições correspondentes

ao índice de cada um dos objetos eleitos como medóide dos grupos. A função de adequabilidade adotada foi a inversa da soma das distâncias entre os objetos e os medóides dos seus respectivos grupos. A roleta [Fogel, 1999] foi adotada como operador de seleção. Como operador de recombinação, utilizou-se o método chamado recombinação de subconjuntos misturados (do inglês *mix subset recombination*), que consiste na seleção de um genótipo filho $\mathbf{G}_f$ como uma amostra de tamanho k retirada a partir do conjunto $\{\mathbf{G}_p^1 \cup \mathbf{G}_p^2\}$ formado pela união dos pais $\mathbf{G}_p^1$ e $\mathbf{G}_p^2$, de forma tal que cada gene $g_{f,i}$ do genótipo $\mathbf{G}_f$ é selecionado aleatoriamente e sem reposição do conjunto $\{\mathbf{G}_p^1 \cup \mathbf{G}_p^2\}$. O operador de mutação utilizado, chamado mutação por troca (do inglês *Flip Mutation*), substitui o valor de cada gene, com dada probabilidade, pelo índice de outro objeto do conjunto de dados. Adicionalmente, o genótipo filho também está sujeito a mutação no processo de recombinação.

Em [Estivill-Castro, 1997] foi proposto um algoritmo genético para agrupamento de dados sob os mesmos modelos de representação genotípica, seleção e mutação que os adotados no trabalho [Lucasius et al., 1993]. Dois novos operadores de recombinação foram definidos: R^3 (Recombinação Aleatória Respeitável - do inglês *Random Respectful Recombination*) e RAR_ω (Recombinação Aleatória Classificativa - do inglês *Random Assorting Recombination*). Dado os pais $\mathbf{G}_p^1$ e $\mathbf{G}_p^2$, o operador de recombinação R^3 consiste em repetir todos os genes do conjunto formado por $\mathbf{G}_p^1 \cap \mathbf{G}_p^2$ e selecionar aleatoriamente $k - |\mathbf{G}_p^1 \cap \mathbf{G}_p^2|$ genes do conjunto $\mathbf{G}_p^1 \cup \mathbf{G}_p^2 - \mathbf{G}_p^1 \cap \mathbf{G}_p^2$, formando assim um descendente; no RAR_ω um filho $\mathbf{G}_f$ é formado iterativamente, de forma tal que sorteia-se um valor δ e, se $\delta \leq limiar$, então um gene é escolhido aleatoriamente de $\mathbf{G}_p^1 \cap \mathbf{G}_p^2 - \mathbf{G}_f$; caso contrário, um gene é escolhido de $\mathbf{G}_p^1 \cup \mathbf{G}_p^2 - [(\mathbf{G}_p^1 \cap \mathbf{G}_p^2) \cup \mathbf{G}_f]$. São realizados k sorteios até completar o filho. O método de seleção utilizado é a roleta, e definiu-se também um operador de mutação com probabilidade muito baixa que substitui um medóide por um outro objeto do mesmo grupo.

Em [Sheng & Liu, 2004] foi definido um algoritmo genético para o problema de agrupamento de dados sob os mesmo moldes do trabalho [Lucasius et al., 1993], utilizando seleção por torneio [Davis, 1996] e um algoritmo de busca local que consiste em avaliar a substituição, para cada grupo, de todos os seus objetos como medóide, selecionando como medóides os objetos que atingem o melhor valor para a função de adequabilidade.

Em [Fränti et al., 1997] tratou-se o problema como a determinação dos centróides dos conjuntos, otimizando a função da inversa da soma das distâncias entre os objetos e os centróides dos seus respectivos grupos. Utilizou-se como representação do genótipo uma matriz real $\rho \times k$, sendo ρ as dimensões dos objetos $\mathbf{x}_i$ do conjunto $\mathbf{X}$ de dados, $i = \{1, \ldots, N\}$, e k o número de grupos, cada coluna da matriz representando portanto a posição do centróide de cada um dos grupos no espaço $\Re^\rho$. Diversas estratégias de seleção foram testadas pelos autores e apontaram para a seleção determinística ($\mu + \lambda$) [Bäck et al., 2000] como uma boa opção em alternativa à seleção por roleta. Diversos operadores de recombinação também foram avaliados. Mais especificamente, foram avaliados cinco

operadores de recombinação, sendo um não sensível ao contexto e os demais quatro sensíveis a contexto. Os resultados dos experimentos indicaram que operadores sensíveis ao contexto são mais adequados ao problema de agrupamento de dados. Adicionalmente, foi utilizado o algoritmo das k-médias [Everitt et al., 2001] como busca local.

Em [Maulik & Bandyopadhyay, 2000] foi proposto um algoritmo genético que trata o problema de agrupamento de dados de forma similar ao trabalho [Fränti et al., 1997] quando consideradas a função de adequabilidade e a representação genotípica, diferenciando-se do anterior principalmente pelos operadores de recombinação e mutação: como operador de recombinação foi utilizada a recombinação de um ponto; como operador de mutação é adicionado um pequeno ruído aos genes em um dado intervalo.

No trabalho [Kivijärvi et al., 2003] foi proposto um algoritmo genético com auto-adaptação dos parâmetros. O genótipo foi representado por uma tupla $l = (\omega_l, \gamma_l, \psi_l, \nu_l)$, sendo: ω_l a codificação de cada possível solução como uma estrutura composta por um matriz $\rho \times k$ com as informações dos centróides (onde ρ é o número de atributos e k o número de grupos) e um vetor de N posições com a informação do grupo ao qual cada objeto pertence, onde a i-ésima posição armazena o rótulo do grupo ao qual pertence o i-ésimo objeto; γ_l representa o método de recombinação utilizado; ψ_l a probabilidade de ω_l sofrer mutação e ν_l armazena o módulo do ruído máximo de mutação para ω_l. Os métodos de recombinação utilizados são os descritos por [Fränti et al., 1997]. O método de mutação é dividido em duas etapas: a primeira consiste na substituição de cada centróide da solução por um objeto do conjunto de dados com probabilidade ψ_l, e a segunda etapa consiste em adicionar um ruído (vetor aleatório) para cada centróide. A cada geração os valores de parâmetros $(\gamma_l, \psi_l, \nu_l)$ de um filho são decorrentes do cruzamento dos pais. No entanto, os parâmetros também estão sujeitos a sofrerem mutação com uma dada probabilidade arbitrada. Para γ_l, a mutação corresponde a selecionar aleatoriamente, com distribuição uniforme, um dos métodos de recombinação. Para ψ_l e ν_l a mutação consiste em sortear novos valores com distribuição Gaussiana restritos a $[\psi_{min}, \psi_{max}]$ e $[\nu_{min}, \nu_{max}]$, respectivamente. Após cada aplicação do ruído o genótipo passa por um ajuste fino aplicando algumas iterações do algoritmo das k-médias. Foram avaliadas duas estratégias de seleção: a roleta e a estratégia de seleção determinística ($\mu + \lambda$), onde μ pais serão cruzados entre si gerando $\lambda = (\mu^2 - \mu)/2$ filhos.

Foi proposto em [Krishna & Murty, 1999] um algoritmo genético para otimizar os centróides iniciais para o algoritmo das k-médias [Everitt et al., 2001], com k definido *a priori*. Os genótipos foram codificados como um vetor de inteiros de N posições, sendo a i-ésima posição correspondente ao grupo ao qual pertence o i-ésimo objeto na base de dados. Como método de seleção foi utilizada a roleta, como função de adequabilidade utilizou-se a inversa da soma das distâncias Euclidiana entre todos os objetos e o centróide de seus respectivos grupos. Foi proposto um operador de mutação que

tem maior probabilidade de mutar genes que estejam mais próximos ao centróide de outro grupo. Adicionalmente, ao final de cada geração, é aplicada uma iteração do algoritmo das k-médias, sendo que, quando este resulta em genótipos inválidos (partições com menos de k grupos), são criados subconjuntos unitários (*singletons*) aleatoriamente até atingir o número pré-definido de grupos.

Nos trabalhos [Lu et al., 2004a] e [Lu et al., 2004b] são apresentadas extensões a [Krishna & Murty, 1999], tratando da sensibilidade do algoritmo ao gerar genótipos inválidos e utilizando operadores genéticos sensíveis ao contexto de agrupamento de dados.

De forma geral, os algoritmos discutidos anteriormente têm sua relevância principal nos diversos operadores sensíveis ao contexto projetados. Em termos de função de adequabilidade, a maioria deles utilizam funções baseadas na inversa da soma das distâncias entre os objetos e os protótipos, sendo os protótipos pertinentes a centróides ou medóides, dependendo da definição. Os algoritmos que otimizam medóides [Lucasius et al., 1993, Estivill-Castro, 1997, Sheng & Liu, 2004] podem representar iniciativas bastante promissoras, uma vez que algoritmos clássicos baseados em medóides são em geral de custo quadrático [Kaufman & Rousseeuw, 1990].

2.2 Algoritmos evolutivos aplicados ao problema de agrupamento de dados com estimação automática do número de grupos k

Em [Casillas et al., 2003] foi definido um algoritmo genético que trata o problema de agrupamento de dados como um grafo com N nós, sendo todos interligados entre si por arestas de peso igual à distância Euclidiana entre os objetos os quais os nós representam. Neste trabalho, busca-se calcular a árvore geradora mínima para o grafo e realizar o agrupamento retirando arestas da árvore, tal que, para se obter uma partição de k grupos, basta retirar $k-1$ arestas. A representação genotípica adotada foi um vetor binário de $N-1$ posições, sendo que a i-ésima posição do vetor corresponde à existência da aresta entre o nó (objeto) i e o nó $i+1$, de forma tal que o valor *um* '1' no vetor corresponde à inexistência de conexão entre os nós e o valor *zero* '0' corresponde à existência de uma aresta ligando os dois nós. Como função de adequabilidade foi utilizado o Critério da Taxa de Variância [Casillas et al., 2003] (do inglês *Variance Ratio Criterion*) definida como a razão entre a dissimilaridade média dos objetos entre grupos sobre a dissimilaridade média dos objetos e o centróide de seu respectivo grupo. Como método de seleção foi utilizada a roleta. O operador de recombinação uniforme [Fogel, 1999] foi adotado. O operador de mutação adotado, com baixa probabilidade de mutação, substitui uma posição do genótipo por *zero* ou *um*.

Foi proposto em [Liu et al., 2004] um algoritmo genético baseado na técnica de agrupamento do

vizinho mais próximo. A geração inicial é obtida a partir do pré-processamento da base de dados, aplicando o método do vizinho mais próximo, de custo $O(N^2)$. Neste método, cada grupo é caracterizado pelos seus objetos serem vizinhos mais próximos e pela distância entre um objeto e o seu vizinho mais próximo não superar um dado limiar. Gera-se uma partição inicial de v grupos. O papel do algoritmo genético nesse trabalho consiste em selecionar k' grupos do conjunto de v grupos, de forma que os grupos selecionados representam a partição dos dados e os não selecionados são considerados ruído, o que no contexto de detecção de intrusos (foco daquele trabalho) indica intrusos na rede. Cada genótipo é representado como um vetor binário de tamanho v, onde cada posição do vetor corresponde à seleção de um grupo. O método de seleção foi baseado no algoritmo da têmpera simulada (do inglês *simulated annealing*), onde cada genótipo com função de adequabilidade acima de um dado limiar é sempre escolhido e os demais são ou não escolhidos em função de alguma probabilidade P, decrescente em razão do número de gerações. Os operadores de recombinação e mutação definidos foram respectivamente a recombinação de um ponto e a escolha aleatória de uma posição do vetor cujo valor será substituído por *zero* ou *um*, dependendo do valor anterior. Como função objetivo foi utilizada uma razão entre a média das distâncias inter-grupos sobre a média das distâncias intra-grupo. Este trabalho foi desenvolvido para aplicação em detecção de intrusos em uma rede e, por esse problema possuir características bastante peculiares, não é facilmente aplicável a problemas que não sigam a mesma característica de distribuição de objetos.

Em [Bandyopadhyay & Maulik, 2002] foi proposto um algoritmo genético que otimiza o índice de Davies-Boldin [Davies & Bouldin, 1979]. Nesse algoritmo a representação do genótipo corresponde a um vetor de tamanho k_{max}, em que k_{max} indica o número máximo de grupos. Cada posição do vetor corresponde às coordenadas do centróide do grupo o qual a posição representa. Posições sem grupos definidos usam um símbolo especial para indicar a ausência de grupo naquela dada posição do vetor. Como operador de recombinação foi utilizada a recombinação de um ponto e como operador de mutação é adicionado um número aleatório com distribuição uniforme.

[Cowgill et al., 1998] propuseram um algoritmo genético em que os genótipos foram codificados como um vetor de inteiros de N posições, sendo a i-ésima posição correspondente ao grupo ao qual pertence o i-ésimo objeto na base de dados. Como função de adequabilidade foi utilizada o Critério da Taxa de Variância [Casillas et al., 2003]. Como método de seleção foi utilizado o *Ranking* [Fogel, 1999] e como operador de recombinação foi utilizada a recombinação de um ponto. Como mutação é escolhido probabilisticamente um gene e seu valor substituído por um número aleatório. Ao final das gerações é aplicado o algoritmo das k-médias a fim de prover uma solução mais refinada.

Em [Hruschka & Ebecken, 2003] foi desenvolvido o *CGA* (*Clustering Genetic Algorithm*) um algoritmo genético para solução do problema de agrupamento de dados. As soluções foram codificadas nos genótipos como um vetor de inteiros de tamanho $N + 1$, onde o i-ésimo elemento do vetor

corresponde ao grupo ao qual pertence o i-ésimo objeto do conjunto de dados e a posição $N+1$ corresponde ao número total de grupos da solução. A função de adequabilidade utilizada foi a Silhueta [Kaufman & Rousseeuw, 1990]. O operador de seleção é a roleta + elitismo. O operador de recombinação utilizado, sensível ao contexto, sorteia m grupos de um dos pais, $\mathbf{G}_p^1$, onde $(1 \leq m \leq k_1)$ e k_1 é o número total de grupos de $\mathbf{G}_p^1$, e copia esses grupos para o outro pai, $\mathbf{G}_p^2$. Os grupos que não forem afetados de $\mathbf{G}_p^2$ serão mantidos e os que forem afetados terão seus elementos reagrupados para os grupos mais próximos (entre os não afetados e os oriundos do pai $\mathbf{G}_p^1$). Foram definidos dois operadores de mutação. O primeiro elimina um dos grupos (aleatoriamente) do genótipo e redistribui seus elementos para os demais grupos; o segundo divide aleatoriamente um grupo em dois. O custo computacional do algoritmo é superiormente limitado pela função de adequabilidade como $O(N^2)$.

Em [Hruschka et al., 2004b] [Hruschka et al., 2004c] [Hruschka et al., 2006] foram apresentadas extensões ao trabalho do *CGA*, simplificando a função objetivo à $O(N)$, que é a componente mais dispendiosa do algoritmo, e empregando o algoritmo das k-médias como operador de busca local. Em [Hruschka et al., 2004c, Hruschka et al., 2006] os autores propuseram operadores de mutação mais atuantes (agressivos) e retiram o operador de recombinação, o componente do algoritmo genético mais pesado computacionalmente, gerando assim o algoritmo chamado Algoritmo Evolutivo para Agrupamento de Dados (*EAC*).

O algoritmo *EAC* representa um bom compromisso entre simplicidade de representação e implementação, ordem computacional e eficácia dos operadores (sensíveis ao contexto), e será discutido em detalhes no Capítulo3.

2.3 Resumo

Neste capítulo apresentou-se algumas das principais abordagens evolutivas aplicadas a agrupamento de dados. Embora existam diversas ferramentas voltadas para esse fim, poucas delas são capazes de otimizar automaticamente o número de grupos. Dentre essas, o *EAC* é uma ferramenta de particular importância tanto por sua simplicidade quanto pela eficiência. Tal algoritmo será descrito mais detalhadamente na seção 3.1 por se tratar da base para os algoritmos que serão propostos nesta dissertação.

Capítulo 3

Algoritmo evolutivo rápido para agrupamento de dados *Fast- EAC*

Um novo algoritmo denominado Algoritmo Evolutivo Rápido para Agrupamento de Dados (*Fast-EAC*) será introduzido neste capítulo. Tal ferramenta é uma extensão do algoritmo *EAC* proposto originalmente em [Hruschka et al., 2004c], empregando diversas modificações nesse algoritmo no intuito de melhorar seu desempenho em termos de eficiência computacional. Nesse contexto, cinco novos operadores evolutivos são incorporados ao *EAC*, resultando em cinco novos algoritmos evolutivos para agrupamento de dados. Desses operadores, aqueles que constituírem melhorias significativas em relação ao *EAC* original serão incorporados ao novo Algoritmo Evolutivo Rápido para Agrupamento de Dados (*Fast-EAC*).

3.1 Algoritmo *EAC*

O *EAC* é baseado em um esquema de codificação bastante simples. Para explicá-lo, considere um genótipo $\mathbf{G}_i$ constituído por um vetor de N inteiros que codifica uma partição de um conjunto de dados com N objetos. Cada posição do vetor corresponde a um objeto, de forma que a i-ésima posição do vetor armazena o rótulo do grupo ao qual pertence o i-ésimo objeto. Conseqüentemente, cada componente do genótipo possuirá um valor de rótulo de grupo $(1, 2, \ldots, k)$. Na Figura 3.1 é ilustrado o esquema de codificação assumindo um conjunto de dados contendo dez objetos particionados em três grupos ($N = 10$ e $k = 3$). As posições correspondentes dos objetos em relação à ordem de aparecimento na base de dados seguem abaixo do genótipo. Neste caso, quatro objetos (1, 2, 7, 8) formam o grupo rotulado com '3'. O grupo rotulado com '1' é formado por três objetos (4, 6, 9) e assim por diante.

Operadores genéticos tradicionais geralmente não são adequados para problemas de agrupamento

de dados por diversas razões, conforme discutido em [Falkenauer, 1998]. No contexto da representação anterior, um primeiro problema em aplicar operadores tradicionais é que o esquema de codificação empregado é inerentemente redundante. Em decorrência de não se aplicar qualquer tipo de restrição acerca da ordem de apresentação dos rótulos dos objetos, um mesmo grupo pode ser codificado de forma diferente bastando que seus objetos tenham o mesmo rótulo, não importando qual ele seja. Especificamente, a codificação adotada pelo *EAC*, ilustrada na Figura 3.1, tem redundância na ordem de $k!$. Por exemplo, uma dada partição de $N = 6$ objetos e $k = 3$ grupos codificada por um genótipo $\mathbf{G}_1$ pode ser codificada por outros cinco genótipos representando a mesma solução, conforme apresentado na Figura 3.2: Desta forma, ao adotar esse tipo de codificação, tem-se que o espaço de busca é ampliado em proporção a $k!$, comprometendo possivelmente a eficiência computacional do algoritmo evolutivo. Devido a essa redundância, operadores de recombinação tradicionais podem apresentar um efeito indesejável ao manipular dois genótipos pais que representem partições idênticas, codificadas de formas diferentes. Especificamente, é esperado que ao combinar dois pais idênticos com um operador de recombinação obtenha-se filhos idênticos aos pais. Essa propriedade não é observada nesta codificação. No exemplo a seguir os genótipos $\mathbf{G}_1$ e $\mathbf{G}_3$ que codificam a mesma partição são submetidos ao operador de recombinação de um ponto. Na Figura 3.3 observa-se que os genótipos descendentes $\mathbf{G}_f$' e $\mathbf{G}_f$" codificam outras partições com $k = 2$ grupos, mas até partições com número distinto de grupos poderiam ser obtidas.

Uma alternativa para evitar esse problema é aplicar algum algoritmo de renumeração nos genótipos. Desta forma evita-se que genótipos distintos representem uma mesma partição dos objetos. Mesmo sendo possível superar essa limitação imposta pela codificação adotada, operadores tradicionais geralmente manipulam os objetos somente pelo rótulo de seus grupos, sem levar em consideração suas conexões com outros grupos. Certamente, as interconexões entre as posições do vetor (dado pelos rótulos dos grupos) constituem o objetivo genuíno de otimização em problemas de agrupamento de dados. Por essa razão, o desenvolvimento de operadores evolutivos especialmente projetados para tarefas de agrupamento de dados tem sido investigado (veja [Falkenauer, 1998]). Nesse contexto, pode-se destacar os operadores do algoritmo *CGA* [Hruschka & Ebecken, 2003], já mencionado na seção 2.2. Os operadores de mutação desse algoritmo constituem a base dos operadores do algoritmo *EAC*, descritos a seguir.

3	3	2	1	2	1	3	3	1	2
1	2	3	4	5	6	7	8	9	10

Figura 3.1: Esquema de codificação do *EAC*.

3.1.1 Operadores de mutação

Dois operadores de mutação são usados no *EAC*. O primeiro operador só pode ser aplicado sobre genótipos que codificam mais do que dois grupos. Ele elimina um ou mais grupos, selecionado-os aleatoriamente e alocando seus objetos nos grupos restantes mais próximos (de acordo com a posição de seus respectivos centróides). Na Figura 3.4 é ilustrado o funcionamento do operador de mutação OM_1. Na representação do conjunto de dados na Figura 3.4 os objetos representados como "círculo" correspondem aos objetos de grupo de rótulo '1', os objetos representados com "losango" correspondem aos objetos de grupo de rótulo '2' e os objetos representados como "estrela" correspondem aos objetos de grupo de rótulo '3', sendo a fronteira dos grupos delineada pelas linhas tracejadas e os centróides dos grupos representados por "x". O Algoritmo 2 provê o pseudo-código deste operador de mutação.

Algoritmo 2 Pseudo-código do operador de mutação OM_1.

1: $[\mathbf{G}_{t+1}]$ = **operador de mutação OM**$_1$(**X**, $\mathbf{G}_t$);
//Seja $\mathbf{G}_t$ um dado genótipo que representa k_G grupos na geração t//
2: **Se** $k_G > 2$ **Então**
3: Gere aleatoriamente um número $m \in \{1, \ldots, k_G - 2\}$; *//pelo menos 2 grupos irão permanecer em $\mathbf{G}_t$//*
4: **Para** $i = (1, \ldots, m)$ **faça**
5: Escolha aleatoriamente um grupos $\mathbf{C}_s$ codificado em $\mathbf{G}_t$;
6: Coloque os objetos pertencentes a $\mathbf{C}_s$ nos grupos mais próximos $\mathbf{C}_j \in \mathbf{G}_t$, $j \neq s$, de acordo com as distâncias entre os objetos e os respectivos centróides;
7: **Fim para**
8: **Senão**
9: $\mathbf{G}_t$ não sofrerá mutação
10: **Fim se**

Para ilustrar o funcionamento do operador de mutação OM_1, considera-se o genótipo $\mathbf{G}_1$ na Figura 3.4(a). Tal genótipo codifica uma partição de $N = 10$ objetos em $k = 3$ grupos. Neste genótipo é selecionado um grupo para sofrer mutação; especificamente, o grupo cujos objetos são rotulados por

$\mathbf{G}_1$: 112233
$\mathbf{G}_2$: 113322
$\mathbf{G}_3$: 331122
$\mathbf{G}_4$: 223311
$\mathbf{G}_5$: 332211
$\mathbf{G}_6$: 221133

Figura 3.2: 6 genótipos representando a mesma partição de dados.

$\mathbf{G}_1$: 11|2233
$\mathbf{G}_3$: 33|1122
Aplicando o operador de recombinação de um ponto
↓
$\mathbf{G}_f$': 111122
$\mathbf{G}_f$": 332233

Figura 3.3: Exemplo de resultado indesejável de operadores de recombinação padrão (insensibilidade ao contexto).

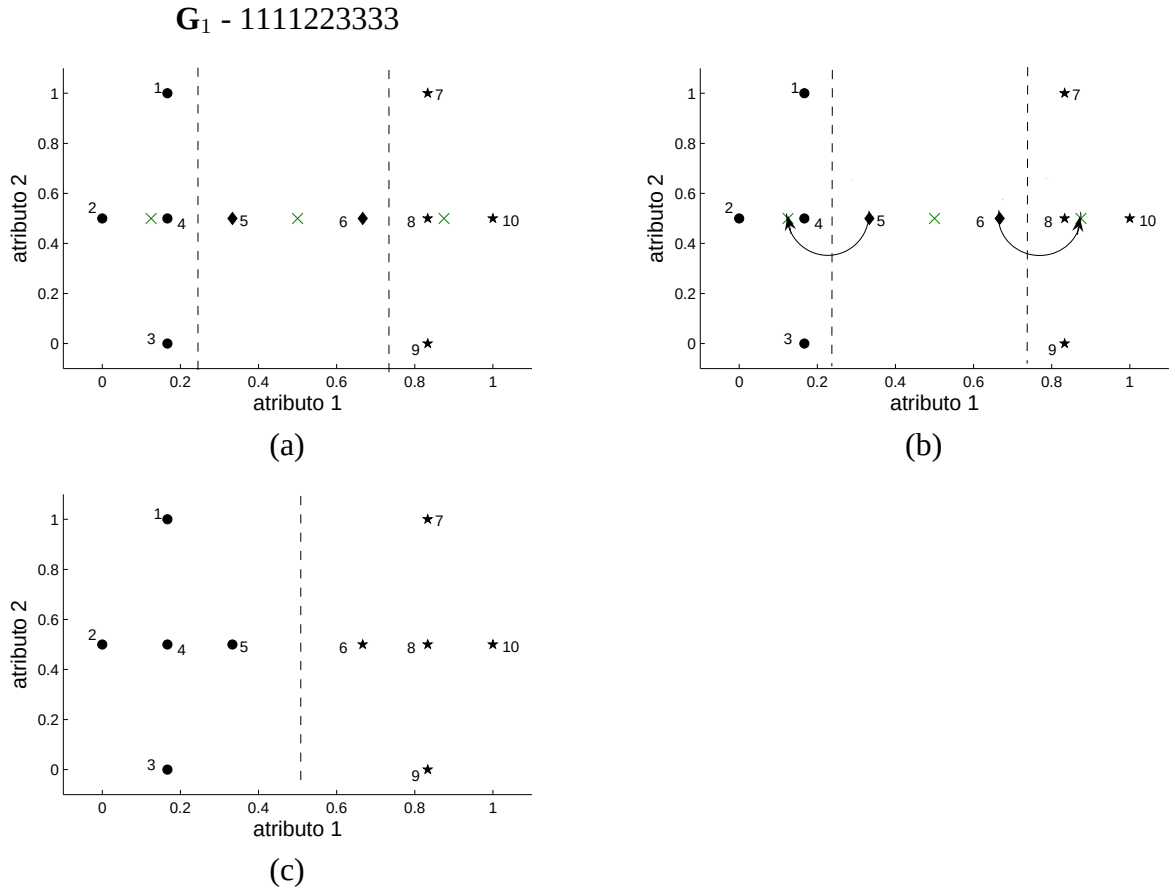


Figura 3.4: Exemplo de aplicação do operador de mutação OM_1: (a) Partição original codificada pelo genótipo; (b) É selecionado um grupo para mutação, sendo este representado pelo rótulo '2'; (c) Genótipo resultante da aplicação do operador de mutação OM_1.

‘2’(losangos). No item (b) da figura são representados os grupos mais próximos de cada um dos objetos e, por fim, no item (c) mostra-se a partição (genótipo) resultante.

O segundo operador de mutação, OM_2, pode ser aplicado somente a grupos formados por dois ou mais objetos. Ele divide um ou mais grupos $(1, \ldots, k_G)$ escolhido(s) aleatoriamente. Cada um dos grupos selecionados é dividido em dois novos grupos ($\mathbf{C}_s$’ e $\mathbf{C}_s$”). Assumindo que um dado grupo $\mathbf{C}_s$ foi escolhido aleatoriamente para mutação, um objeto s_1 de $\mathbf{C}_s$ é selecionado de forma tal que o objeto será usado como semente para gerar o novo grupo $\mathbf{C}_s$’. O objeto s_2 mais distante de s_1 é então selecionado como a semente para gerar o grupo $\mathbf{C}_s$”. Em função dos objetos escolhidos como sementes, os objetos pertencentes a $\mathbf{C}_s$ mais próximos de s_1 são atribuídos ao grupo $\mathbf{C}_s$’, enquanto os objetos mais próximos de s_2 formam o grupo $\mathbf{C}_s$”. O Algoritmo 3 provê o pseudo-código do operador de mutação OM_2.

A Figura 3.5 ilustra um exemplo da aplicação deste operador sobre um genótipo $\mathbf{G}_1$. Na representação do conjunto de dados na Figura 3.5 os objetos representados como “círculo” correspondem aos objetos de grupo de rótulo ‘1’ e os objetos rfepresentados como “losango” correspondem aos objetos de grupo de rótulo ‘2’. A Figura 3.5(a) apresenta o genótipo original que será submetido a mutação pelo operador de mutação OM_2. Na Figura 3.5(b) são selecionadas duas sementes (objetos em destaque), s_1 e s_2. Na Figura 3.5(c) apresenta-se quais objetos do grupo de rótulo ‘1’ são mais próximos de quais semente. Por fim, dois novos grupos são gerados, $\mathbf{C}_s$’ com os objetos mais próximos de s_1 e $\mathbf{C}_s$” com os objetos mais próximos de s_2.

Algoritmo 3 Pseudo-código do operador de mutação OM_2.

1: $[\mathbf{G}_{t+1}]$ = **operador de mutação OM$_2$(X, G$_t$)**
//Seja $\mathbf{G}_t$ um dado genótipo que representa k_G grupos na geração t//
2: Gere aleatoriamente um número $m \in \{1, \ldots, k_G\}$;
3: **Para** $i = (1, \ldots, m)$ **faça**
4: Escolha aleatoriamente um grupo $\mathbf{C}_s \in \mathbf{G}_t$;
5: **Se** $|\mathbf{C}_s| \geq 2$ **Então**
6: Escolha aleatoriamente um objeto $s_1 \in \mathbf{C}_s$;
7: Determine o objeto $s_2 \in \mathbf{C}_s$ mais distante de s_1;
8: Gere dois novos grupos $\mathbf{C}_s$’ e $\mathbf{C}_s$”, alocando os objetos de $\mathbf{C}_s$ mais próximos a s_1 para o grupo $\mathbf{C}_s$’ e os demais para o grupo$\mathbf{C}_s$”;
9: **Senão**
10: Não divida o grupo $\mathbf{C}_s$;
11: **Fim se**
12: **Fim para**

Com os operadores de mutação OM_1 e OM_2 descritos acima, alguns genótipos (soluções candidatas) podem ser pouco modificados, com poucos grupos afetados (em torno de um ou dois), enquanto outros podem ser extremamente modificados, com muitos grupos afetados, significando um grande

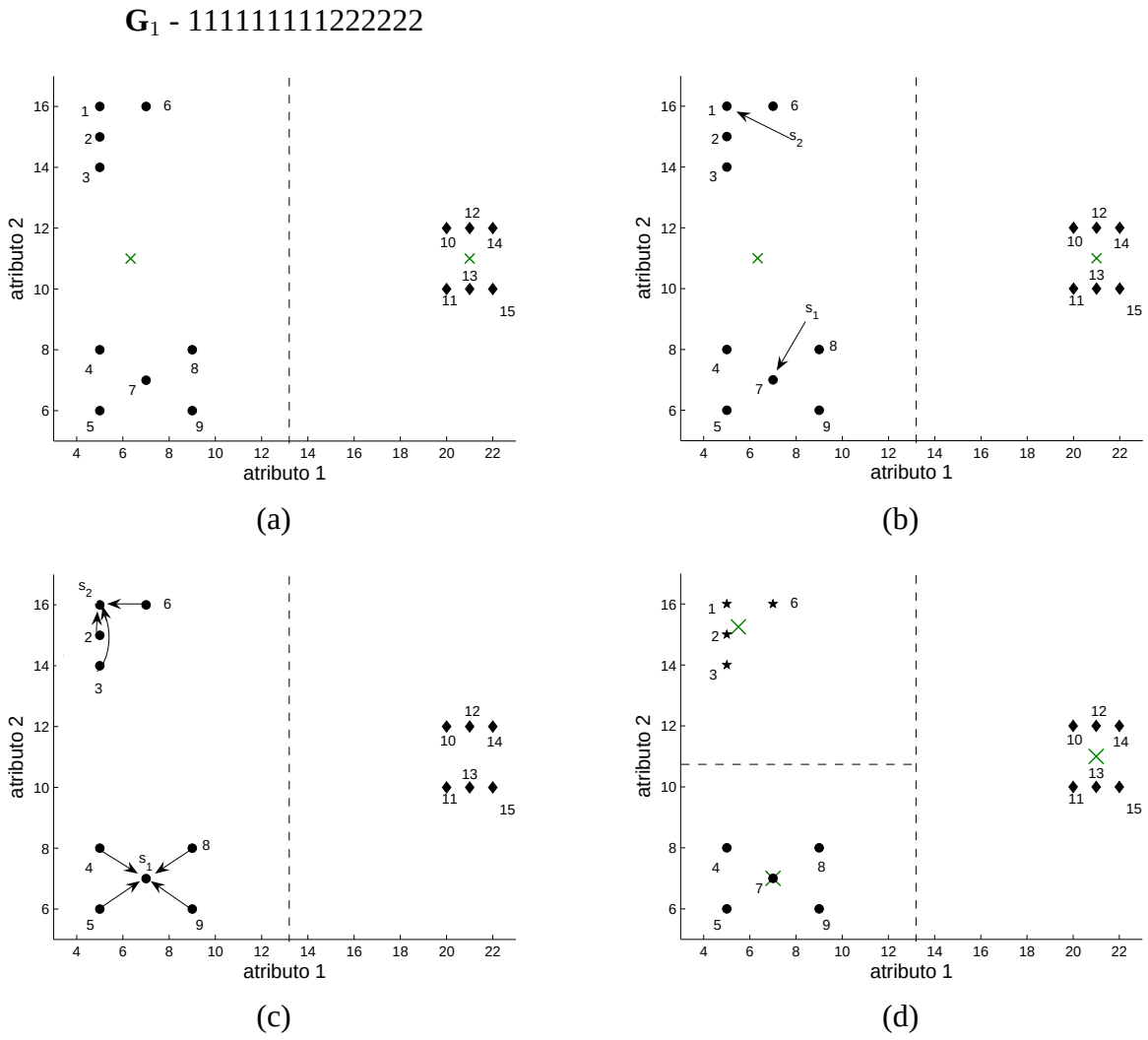


Figura 3.5: Exemplo de aplicação do operador de mutação OM_2: (a) Partição original codificada pelo genótipo; (b) É selecionado um grupo para mutação, sendo este representado pelo rótulo ‘1’; (c) Indica-se quais objetos do grupo ‘1’ são mais próximos de quais sementes; (d) Genótipo resultante da aplicação do operador de mutação OM_2.

salto no espaço de busca. Esta é uma propriedade desejada para algoritmos evolutivos, pois contribui para manter a diversidade populacional, possibilitando que esse fique menos suscetível a ótimos locais.

Conforme definido originalmente em [Hruschka et al., 2004c, Hruschka et al., 2006], o *EAC* não emprega probabilidades de mutação, ou seja, os operadores definidos são necessariamente aplicados a alguns genótipos selecionados. Particularmente, 50% dos genótipos de uma dada população são mutados pelo OM_1 e os demais são mutados pelo OM_2, garantindo assim cobertura total dos genótipos e equilíbrio entre os processos de aumento e decaimento de grupos.

3.1.2 Busca local pelo algoritmo das k-médias

O *EAC* emprega o popular algoritmo das k-médias como um processo de busca local, o qual desempenha um ajuste fino nas partições obtidas pelos operadores evolutivos, desta forma acelerando a convergência do *EAC*. Algoritmos de agrupamento de dados envolvendo o cálculo de vetores médios (centróide) de cada um dos grupos são freqüentemente referenciados como algoritmos de k-médias [Everitt et al., 2001]. Basicamente, estes algoritmos particionam um conjunto de dados de N objetos em k grupos, tentando minimizar a soma das distâncias entre os objetos de um grupo e seu respectivo centróide. O *EAC* faz uso do 'k-médias' descrito no Algoritmo 4. O critério de parada pode ser definido tanto como um número máximo de iterações (t) do Passo 3 ou como a diferença máxima entre os centróides em duas iterações consecutivas.

Algoritmo 4 Pseudo-código do algoritmo k-médias.

1: [Partição] = **algoritmo** k**-médias**(**X**, k)
2: Gere a partição inicial (k grupos não vazios);
3: **Enquanto** o critério de parada não é satisfeito **faça**
4: Calcule o centróides dos grupos (vetores médios) da partição corrente;
5: Coloque cada objeto no grupo com o centróide mais próximo;
6: **Fim enquanto**

Existe uma sinergia entre o k-médias e os operadores do *EAC*. Por um lado, o *EAC* pode minimizar dois problemas principais do k-médias, a saber: (i) k-médias pode ficar preso em centróides subótimos (mínimos locais da função objetivo); e (ii) o número de grupos (k) precisa ser definido *a priori*. Como os operadores do *EAC* podem eliminar, dividir, e juntar grupos por meio da busca evolutiva, eles são capazes de evoluir para partições melhores em termos tanto de número de grupos quanto de centróides. Essas partições podem fornecer centróides iniciais melhores para o k-médias, assim aumentado a probabilidade do k-médias convergir para um ótimo global. Por outro lado, o k-médias minimiza a variância dos grupos modificados pelos operadores do *EAC*. Adicionalmente,

quando k é superestimado, o algoritmo das k-médias pode reduzir convenientemente o número de grupos. Isso porque irá "eliminar" todo grupo $\mathbf{C}_i$, representado por um centróide $\mathbf{v}_i$, para o qual não exista pelo menos um objeto que seja mais próximo de $\mathbf{v}_i$ do que de um outro centróide $\mathbf{v}_j$ de um grupo qualquer $\mathbf{C}_j$, $i \neq j$. Essa característica também ajuda os operadores do *EAC* a concentrar esforços nas regiões mais promissoras do espaço de busca, reduzindo k quando apropriado de forma mais rápida do que a busca evolutiva desprovida desse mecanismo. Descrições mais detalhadas dos efeitos do algoritmo das k-médias sobre o processo de busca evolutiva do EAC podem ser encontradas em [Hruschka et al., 2004c, Hruschka et al., 2006].

3.1.3 População inicial, seleção e passos principais

Nos experimentos descritos em [Hruschka et al., 2004c, Hruschka et al., 2006], a população inicial do *EAC* é gerada aleatoriamente, de forma tal que cada gene do genótipo receba um valor do conjunto $\{1, 2, \ldots, k\}$ com distribuição de probabilidade uniforme. Pode-se verificar que esta inicialização tende a fazer com que os centróides da população inicial localizem-se aproximadamente na grande média do conjunto de dados. Desta forma, o problema se torna mais difícil ao forçar a otimização a partir de um conjunto de centróides localizados na mesma região do espaço de objetos, diminuindo assim a probabilidade de que a partição inicial consiga representar de forma adequada alguns grupos. Isso é de particular interesse quando se deseja colocar à prova o algoritmo frente a um cenário de aplicação bastante desfavorável. Na prática, no entanto, uma forma mais eficiente e simples de inicialização é sortear as posições dos centróides iniciais dentro do domínio de valores dos atributos e atribuir cada objeto ao centróide mais próximo.

Quanto ao número de grupos, o usuário pode definir, para cada genótipo, tanto o valor inicial para k ou sortear esses valores dentro de valores mínimos e máximos de um dado intervalo. Valores superestimados e subestimados para k podem ser adotados já que o *EAC* pode diminuir ou aumentar este número inicial de grupos em direção a uma melhor estimativa (de acordo com a informação fornecida pela função de adequabilidade). A função de adequabilidade originalmente empregada pelo *EAC* é a versão simplificada [Hruschka et al., 2004b, Hruschka et al., 2004c] da silhueta proposta em [Kaufman & Rousseeuw, 1990]. Essa função é descrita na seção 3.3.2.

Uma estratégia elitista, na qual o melhor genótipo (o de maior adequabilidade) é sempre copiado e mantido na geração seguinte [Fogel, 1999], é utilizada no *EAC*. Cada genótipo da população tem seu valor de adequabilidade normalizado linearmente [Davis, 1996] e os genótipos para a próxima geração são selecionados de acordo com a estratégia da roleta (seleção proporcional) [Eiben & Smith, 2003]. Os principais passos do *EAC* são representados no Algoritmo 5.

De um ponto de vista prático, o critério de convergência do algoritmo pode ser estabelecido por diversas formas; por exemplo, um número pré-determinado de gerações sem qualquer melhora

Algoritmo 5 Pseudo-código para o Algoritmo Evolutivo para Agrupamento de Dados (*EAC*).

1: [Partições] = **algoritmo *EAC***(**X**, T)
2: Gere a população inicial aleatoriamente;
3: **Enquanto** o critério de convergência não é satisfeito **faça**
4: Aplique o algoritmo das k-médias a todos os genótipos;
5: Avalie cada genótipo de acordo com a função de adequabilidade;
6: Aplique normalização linear;
7: Selecione os genótipos para mutação (com reposição) pela Roleta;
8: Aplique os operadores de mutação; *//50% dos genótipos selecionados com o operador OM_1 e os demais genótipos com o operador OM_2//*
9: Substituir os genótipos antigos pelos formados no Passo 8;
10: **Fim enquanto**

no valor da função de adequabilidade ou a diversidade populacional abaixo de um dado limiar [Eiben & Smith, 2003]. Nas avaliações empíricas realizadas no presente trabalho será adotado como critério de convergência um valor meta de referência para a função de adequabilidade.

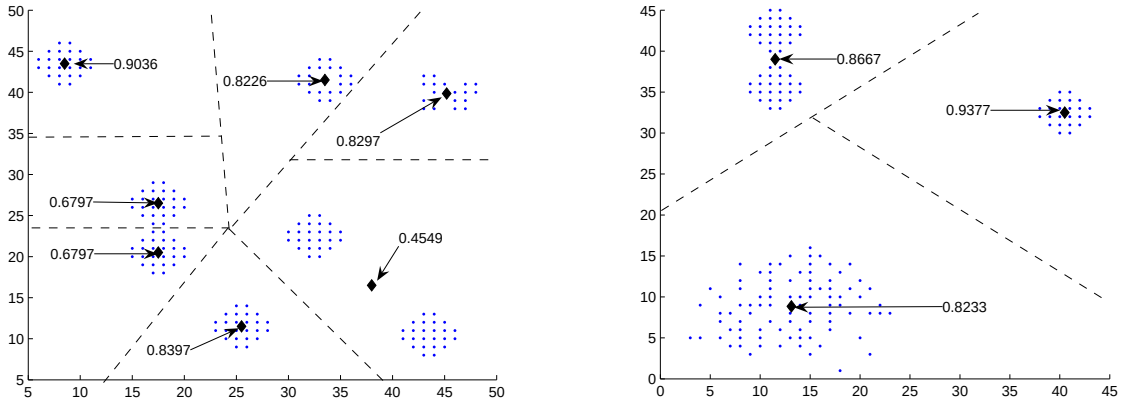
3.2 Melhorias propostas para o *EAC*

3.2.1 *EAC*-I

Como detalhado na seção 3.1, os operadores de mutação do *EAC* foram projetados para dividir e eliminar grupos selecionados aleatoriamente. Alternativamente, é possível usar a informação sobre a qualidade dos grupos para selecioná-los para mutação. Como a qualidade de uma partição é mensurada por um critério numérico, cada grupo de uma partição será avaliado individualmente por algum índice de qualidade parcial por grupo. Neste trabalho, a primeira proposta para melhorar o *EAC* segue esse caminho, basicamente modificando a forma que os operadores de mutação do *EAC* selecionam os grupos para mutação (Passo 5 no Algoritmo 2 e Passo 4 no Algoritmo 3). Particularmente, assume-se que melhores grupos devem possuir menor probabilidade de sofrer mutação. Essa hipótese, baseada no senso comum, faz com que bons grupos tendam a ser mantidos durante o processo evolutivo, na esperança de melhorar, via uma polarização probabilística, a partição codificada no genótipo.

Não existem garantias teóricas de que essa heurística poderá apresentar de fato resultados melhores do que desprezando a informação da qualidade dos grupos. Na Figura 3.6 são apresentados dois exemplos, um ilustrando um caso no qual guiar a busca por grupos utilizando a informação de qualidade parcial poderá acelerar a convergência e um outro caso em que o mesmo tende a não ocorrer. Nessa figura, os pontos correspondem aos objetos do conjunto de dados e os losangos aos centróides dos grupos. As fronteiras dos grupos são representadas pelas linhas pontilhadas e os ró-

tulos numéricos correspondem aos valores do critério numérico parcial de cada grupo. Na Figura 3.6(a), o grupo de menor função de adequabilidade parcial (0,4549) possui, por inspeção visual, duas classes, enquanto os demais grupos da partição parecem estar formados adequadamente. Isso, nesse caso, indica que utilizar a informação da função de adequabilidade parcial pode apresentar resultados melhores do que a busca com distribuição uniforme. Na Figura 3.6(b) é apresentado um caso típico em que fazer uso da informação de qualidade de um grupo não representará aumento da probabilidade de aplicar mutação sobre o grupo mais adequado para aquela partição. O *cluster* de pior função de adequabilidade parcial corresponde a um grupo bastante difuso. Entretanto, por inspeção visual não é possível indicar uma forma de estruturar os objetos deste grupo mais adequadamente. Por outro lado, em outro grupo, com o segundo melhor valor de função de adequabilidade, percebe-se claramente que é possível estruturá-lo de forma diferente, eventualmente mais apropriada.



(a) (b)

Figura 3.6: Duas partições. (a) partição adequada à busca guiada; (b) partição não adequada à busca guiada.

Formalmente, considera-se que uma função de adequabilidade (f) que mede a qualidade da partição codificada por um dado genótipo **G** possa ser decomposta em contribuições parciais $f_c(\mathbf{C}_i)$ de cada grupo $\mathbf{C}_i$ ($i = 1, \ldots, k$) codificado em **G**, tal que:

$$\begin{aligned} f(\mathbf{G}) &= \textstyle\sum_{i=1}^{k} f_c(\mathbf{C}_i) \\ f_c(\mathbf{C}_i) &\in [0, 1] \end{aligned} \tag{3.1}$$

A fim de determinar qual tipo de função deveria ser adotada como opção para a seleção informada dos grupos $\mathbf{C}_i$ para mutação, foram definidas duas características entendidas como interessantes para guiar a busca evolutiva em seus diferentes estágios. Estas características podem ser vistas como arbitradas, a serem avaliadas empiricamente.

A seguir seguem descritas as características entendidas como desejáveis para a função de probabilidade:

1. Quando um genótipo codifica uma partição de baixa qualidade é desejável que as probabilidades de mutação não tenham grande variabilidade, evitando desta forma que alguns grupos ruins para uma dada partição sejam privilegiados em detrimento de outros grupos de qualidade inferior. Em outras palavras, se todos os grupos são ruins, eles devem possuir probabilidades similares de mutação.

2. Quando um genótipo codifica uma partição de melhor qualidade e na qual os grupos possuem variância pequena na função de adequabilidade parcial, a função de probabilidade deverá ser capaz de diferenciar quais grupos deverão possuir maior probabilidade de sofrer mutação, evitando que a probabilidade de selecionar um grupo de uma dada partição aproxime-se de uma distribuição uniforme.

Com base nessas características, foram propostas duas funções potencialmente adequadas para mapear os valores de função objetivo como probabilidades de mutação para um dado grupo $\mathbf{C}_i$. Na primeira, ao fazer uso da propriedade de $f_c(\mathbf{C}_i) \in [0, 1]$ na Eq.(3.1) considera-se que a probabilidade de um determinado grupo não sofrer mutação pode ser dada pelo próprio valor de função de adequabilidade, ou seja $\overline{P}(\mathbf{C}_i) = f_c(\mathbf{C}_i)$, calculando a probabilidade de mutação como o complemento da probabilidade de não mutar. Logo, $P(\mathbf{C}_i) = 1 - f_c(\mathbf{C}_i)$. A segunda alternativa é considerar que a probabilidade de mutação está associada à função inversa da função de adequabilidade. Mais especificamente, tem-se:

$$P(\mathbf{C}_i) = \frac{[f_c(\mathbf{C}_i)]^{-1}}{\sum_{j=1}^{k} [f_c(\mathbf{C}_i)]^{-1}}$$

Para avaliar o comportamento das funções de probabilidade descritas acima são apresentados dois diferentes cenários, cada qual representando um conjunto de grupos de um genótipo para o qual a função de adequabilidade cresce linearmente ao longo do processo evolutivo. Mais especificamente, os cenários são definidos como:

- O primeiro cenário corresponde a um genótipo $\mathbf{G}_1$ codificando uma partição com quatro grupos $\mathbf{G}_1 = \{\mathbf{C}_1, \mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4\}$, sendo $\mathbf{C}_1$ um grupo com valor de função de adequabilidade parcial equivalente a 10% da função de adequabilidade total da partição. Os demais grupos $\{\mathbf{C}_2, \mathbf{C}_3, \mathbf{C}_4\}$ são responsáveis por 20%, 30% e 40% respectivamente do valor de função de adequabilidade total. Tal configuração servirá para avaliar o comportamento das funções de probabilidade quando os grupos possuem qualidades distintas entre si. Note que, para valores baixos de função de adequabilidade total, será observada uma configuração próxima daquela do item 1 das características desejáveis descritas anteriormente.

- O segundo cenário objetiva avaliar o comportamento das funções de probabilidade quando um genótipo codifica uma partição com grupos de qualidade similar, particularmente quando o valor da função de adequabilidade total for alto. Nesse cenário, a função de probabilidade deverá ser capaz de diferenciar quais grupos terão maior probabilidade de sofrer mutação. Esse cenário é configurado com um genótipo codificando uma partição com quatro grupos cujos valores da função de adequabilidade parcial equivalem a 23%, 24%, 25% e 26% respectivamente da função de adequabilidade total. Note que, para valores altos de função de adequabilidade total, será observada configuração próxima daquela do item 2 das características desejáveis descritas anteriormente.

É importante destacar que os valores associados a cada grupo nos dois cenários descritos acima podem ser considerados arbitrários, sendo que quaisquer valores poderiam ser usados desde que fossem escolhidos de modo a refletir as situações as quais se deseja verificar o comportamento das funções de probabilidade.

Na Figura 3.7 é apresentado o comportamento de cada uma das funções de probabilidade nos diferentes cenários definidos. No eixo da abscissa são apresentadas gerações com valor crescente de função de adequabilidade total (*fitness* total) e no eixo das ordenadas o respectivo valor de probabilidade para cada um dos 4 grupos do genótipo.

A Figura 3.7(a) mostra que a função baseada no complemento da probabilidade de não mutar é adequada ao observar as características estabelecidas como desejáveis, por ser capaz de equalizar as probabilidades no início do processo evolutivo (baixos valores de função de adequabilidade) mesmo quando as diferenças de função de adequabilidade entre os grupos é significativa. Na Figura 3.7(b) é apresentado o segundo cenário. A função continua atendendo de forma coerente as premissas adotadas, visto que, mesmo com valores similares de função de adequabilidade parcial para os grupos, esta é capaz de destacar quais grupos devem de fato possuir maior probabilidade de mutação ao final do processo evolutivo. Nas Figuras 3.7 (c) e (d) em que se avalia a função de probabilidade do inverso da função de adequabilidade, é verificado que no item (c) a função de probabilidade mostrou-se adequada às premissas assumidas. No entanto, no item (d) observa-se que a função de probabilidade não foi capaz de contrastar os valores de probabilidade suficientemente para que a busca tivesse tendência diferente de uma distribuição aproximadamente uniforme.

Em função das análises acima, adotou-se a função do complemento da probabilidade de não sofrer mutação. Os valores de probabilidade de não mutar são fornecidos para um processo de seleção utilizando a roleta sem reposição para a tarefa de selecionar os grupos a serem mutados.

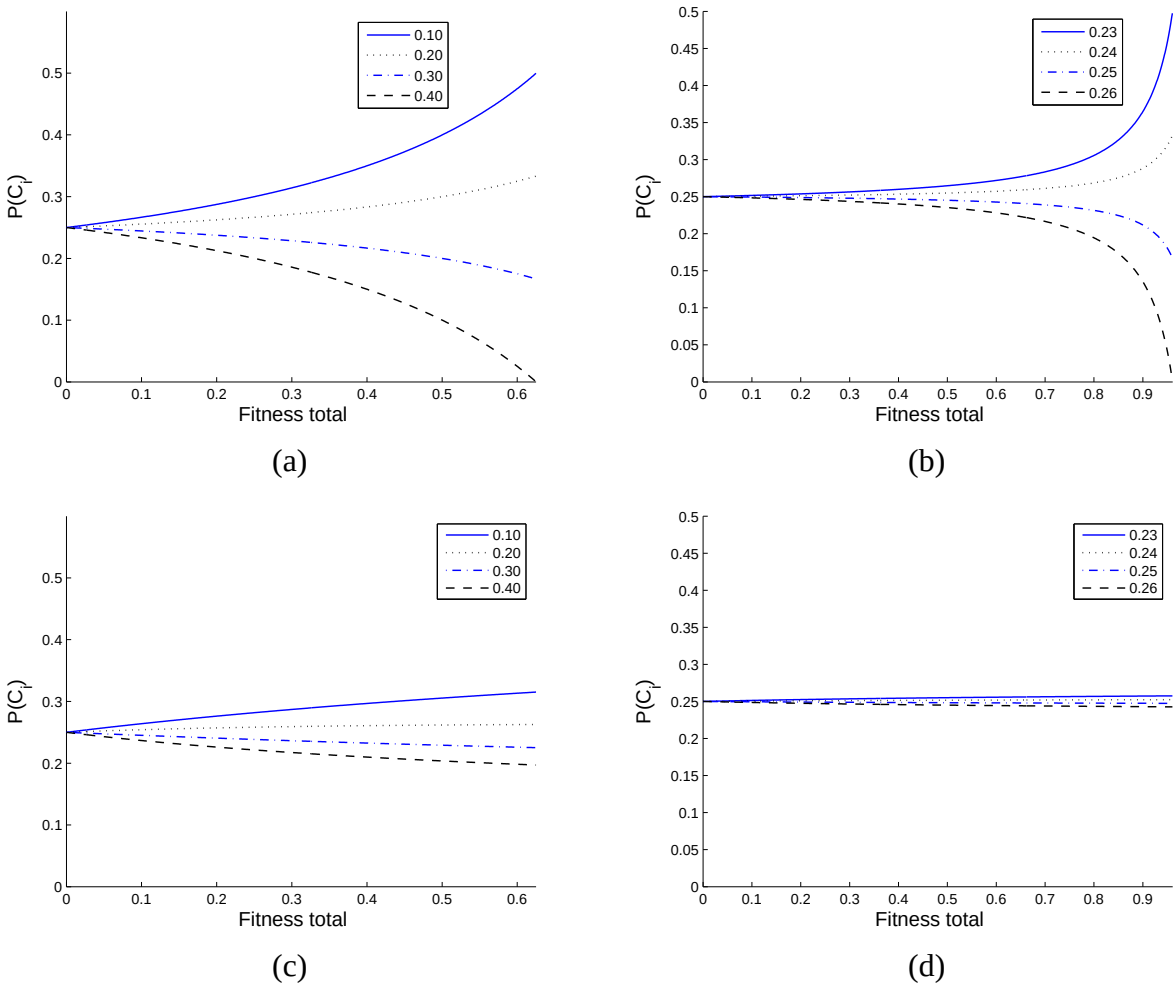
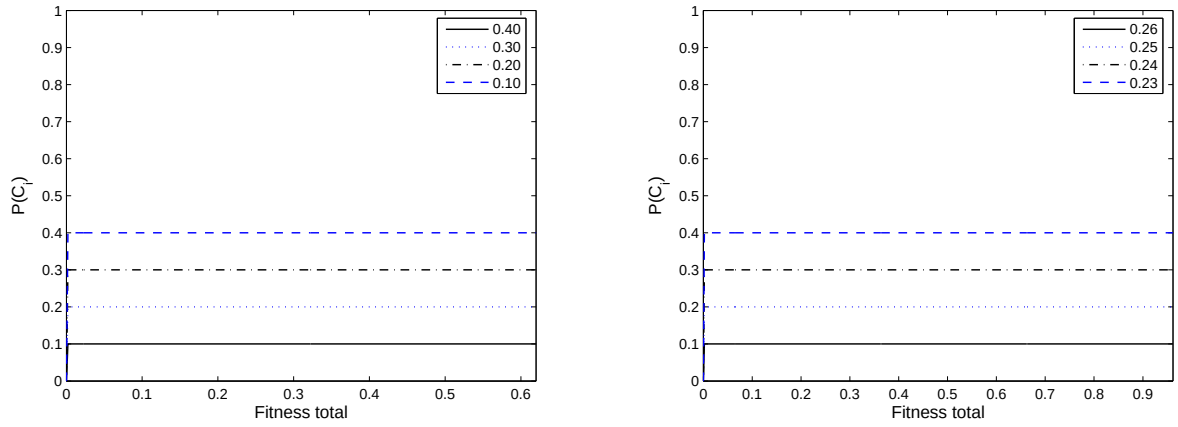


Figura 3.7: Avaliação de funções de probabilidade para seleção de grupos: (a) Função complemento da probabilidade de não mutar, primeiro cenário; (b) Função complemento da probabilidade de não mutar, segundo cenário; (c) Função inverso da função de adequabilidade, primeiro cenário; (d) Função inverso da função de adequabilidade, segundo cenário.

3.2.2 *EAC*-II

A única diferença entre o *EAC*-II e o *EAC*-I é que os valores da função de adequabilidade parcial f_c no *EAC*-II são normalizados de acordo com um processo de normalização linear [Davis, 1996]. É esperado que o processo de normalização linear ajude a evitar problemas de convergência prematura ao balancear a pressão seletiva em diferentes estágios da busca evolutiva. Diferente do *EAC*-I, o *EAC*-II, com a adoção da normalização linear dos valores de função de adequabilidade parcial, terá como característica atribuir probabilidades de mutação para os grupos utilizando a informação de classificação ordinal da qualidade dos grupos. Ao ignorar as diferenças de qualidade absolutas entre os grupos de um dado genótipo, busca-se também criar um algoritmo menos sensível à seleção da função de adequabilidade parcial.

Nas figuras 3.8 (a) e 3.8 (b), que representam a aplicação do primeiro e segundo cenários descritos na seção 3.2.1 respectivamente, percebe-se que ao ignorar as diferenças absolutas de qualidade entre os grupos, de fato reduz-se a influência da distribuição dos valores e faixas de grandeza, tornando desta forma o algoritmo robusto à escolha da função de adequabilidade parcial.



(a) (b)

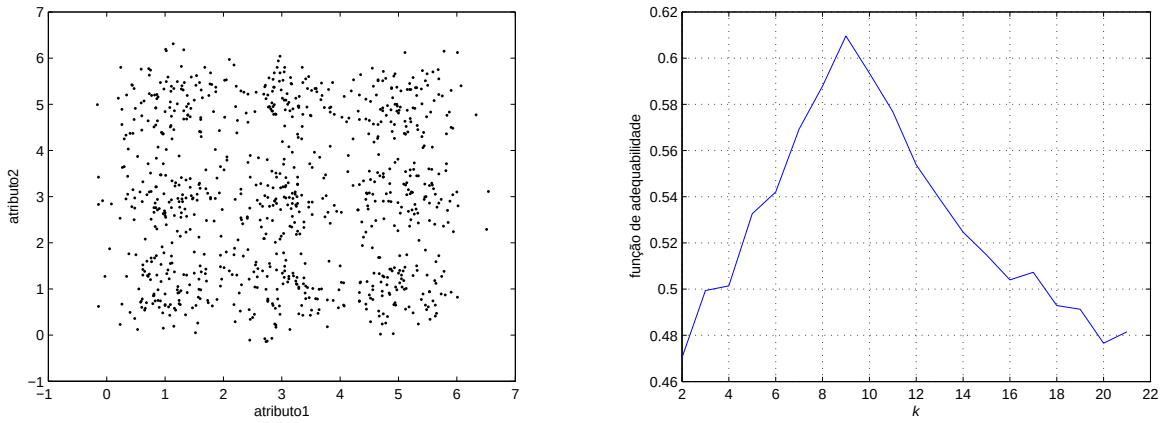
Figura 3.8: Normalização linear dos valores de adequabilidade parcial: (a) Probabilidade de mutar para o primeiro cenário; (b) Probabilidade de mutar para o segundo cenário;

3.2.3 *EAC*-III

No Passo 8 do *EAC* (Algoritmo 5), 50% dos genótipos de uma dada população são mutados pelo operador de mutação OM_1 (Algoritmo 2) e os demais são mutados pelo Operador de mutação 2 (OM_2 - Algoritmo 3). Comumente em algoritmos evolutivos, utilizam-se probabilidades para aplicação dos operadores [Mitchell, 1996]. Em [Kivijärvi et al., 2003], os valores de probabilidades de aplicação

dos operadores evolutivos foram auto-ajustados no processo evolutivo. Da mesma forma, o *EAC*-III foi projetado no intuito de melhorar o desempenho do *EAC* original pelo auto-ajuste da taxa de aplicação dos operadores de mutação. Um mecanismo que seja capaz de perceber o quanto cada operador de mutação contribuiu na melhora do valor médio da função de adequabilidade em uma dada geração poderá ser capaz de ajustar a taxa de aplicação dos operadores de forma que o operador com melhor desempenho seja aplicado mais vezes que o outro.

Na Figura 3.9(b) são ilustrados os respectivos valores de função de adequabilidade máximos, para a base de dados apresentada na Figura 3.9(a), resultantes de diversas partições para cada valor de número de grupos k. A curva representada na Figura 3.9(b) auxilia compreender como a utilização do auto-ajuste das taxas de aplicação dos operadores de mutação pode mostrar-se eficiente. Por exemplo, considerando uma população em que a maioria dos genótipos codificam partições com número superestimado de grupos, nesta instância $k > 9$ grupos, o operador de mutação OM_1, que diminui o número k de grupos, tem maior probabilidade de gerar genótipos com melhor valor de função de adequabilidade e, por conseqüinte, deverá ser aplicado em maior proporção que o OM_2.



(a) (b)

Figura 3.9: (a) Base de dados; (b) Valores de função de adequabilidade para partições com diferentes números de grupos, k.

Formalmente, assumindo que o operador de mutação OM_1 tenha sido aplicado a um número de genótipos $G_{1,t}$ na geração t, pode-se calcular a adequabilidade média do conjunto de $G_{1,t}$ genótipos ($AFOM^a_{1,t}$) antes da aplicação do OM_1 e a adequabilidade média do conjunto $G_{1,t}$ após a aplicação de OM_1 ($AFOM^b_{1,t}$). A diferença entre a média da adequabilidade resultante da aplicação de OM_1 aos genótipos $G_{1,t}$, aqui será referenciada como $DAF_{1,t}$, ou seja, $DAF_{1,t} = (AFOM^a_{1,t} - AFOM^b_{1,t})$. Quantidades análogas podem ser definidas para $MO2$ quando este é aplicado a $T - G_{1,t}$ genótipos, lembrando que T foi definido como a quantidade total de genótipos da população. Dadas as definições

anteriores, as taxas de aplicação de OM_1 e OM_2 na geração $(t+1)$ podem ser calculadas usando as equações (3.2) e (3.3), respectivamente:

$$PMO_{1,t+1} = \begin{cases} \frac{DAF_{1,t}}{DAF_{1,t}+DAF_{2,t}} \; se \; DAF_{1,t} > 0 \wedge DAF_{2,t} > 0 \\ 0 \; se \; DAF_{1,t} \leq 0; \\ 1 \; se \; DAF_{2,t} \leq 0; \\ 0,5 \; se \; DAF_{1,t} \leq 0 \wedge DAF_{2,t} \leq 0; \end{cases} \tag{3.2}$$

$$PMO_{2,t+1} = 1 - PMO_{1,t+1} \tag{3.3}$$

As equações (3.2) e (3.3) mostram que se um operador particular de mutação fornece melhores resultados em uma dada geração t, este será aplicado em maior proporção na geração $(t+1)$. Além disso, se um operador particular não está funcionando, então o outro operador será avaliado e aplicado sobre todos os genótipos. A incorporação deste processo de mutação mais sofisticado no *EAC* original (Algoritmo 5) produz o algoritmo doravante chamado *EAC*-III.

3.2.4 *EAC*-IV

Conforme discutido anteriormente, o operador de mutação OM_1 (Algoritmo 2) do *EAC* original (Algoritmo 5) elimina um ou mais grupos aleatoriamente selecionados, alocando seus objetos no grupo restante mais próximo. Para isso, as distâncias entre cada objeto do grupo a ser eliminado e os centróides de cada um dos grupos restantes são computadas. No *EAC*-IV, uma versão computacionalmente mais eficiente deste operador é empregada. Mais especificamente, qualquer grupo $\mathbf{C}_s$ aleatoriamente selecionado para ser eliminado tem todos os seus objetos alocados em um mesmo grupo $\mathbf{C}_c$ (cujo protótipo é mais próximo do protótipo de $\mathbf{C}_s$). Como o operador OM_1 é projetado para mutar de $\{1, \ldots, k_l - 2\}$ grupos, onde k_l é o número de grupos codificados no genótipo $\mathbf{G}_l$ a ser mutado, isso permite a redução do custo computacional do Passo 6 do OM_1 (Algoritmo 2) de $O(k_l \times N)$ para $O(k_l^2)$, onde N é o número de objetos do conjunto de dados.

Em termos gerais, como o genótipo para o *EAC* é definido como um vetor de inteiros de N posições, qualquer operador evolutivo que opere sobre o genótipo modificando-o no âmbito dos grupos, seja dividindo, juntando ou unindo grupos, tem sua complexidade computacional de tempo inferiormente limitada à $O(N)$. Na Figura 3.10 é apresentado como exemplo o genótipo $\mathbf{G}_1$. Observe que para alterar qualquer grupo codificado em tal genótipo é preciso realizar uma busca do primeiro ao último gene para garantir que todos os objetos pertencentes a determinado grupo sejam alterados.

No *EAC*-IV, para superar essa limitação de complexidade computacional imposta pela codificação do genótipo, utilizou-se uma estrutura adicional contendo informações sumarizadas para cada grupo.

Essas informações possibilitam que o *EAC*-IV opere diretamente nos centróides dos grupos, ao invés de realizar tal ação indiretamente ao operar nos objetos (genes) da partição (genótipo). Especificamente, a estrutura adicional é definida como uma matriz $k \times (\rho + 2)$, onde cada linha da submatriz $k \times \rho$ mais à esquerda corresponde à posição no espaço ρ-dimensional do centróide de cada um dos k grupos codificados no genótipo e as duas colunas restantes correspondem aos respectivos valores de rótulo do grupo e número de objetos que pertencem ao mesmo. Na Tabela 3.1 é apresentado um exemplo da estrutura auxiliar. As colunas n_1 e n_2 correspondem às posições dos centróides em cada uma das suas duas dimensões. A coluna "rótulo" corresponde ao rótulo do grupo o qual o centróide representa e a coluna "objetos" corresponde ao número de objetos pertencentes àquele grupo. Especificamente, a Tabela 3.1 refere-se à partição apresentada na Figura 3.11(a).

Tabela 3.1: Estrutura auxiliar baseada em centróides utilizada pelo *EAC*-IV.

n_1	n_2	rótulo	objetos
17	8,84	1	44
10	8,84	2	45
11,5	35	3	24
40,5	32	4	24
11,5	42	5	24

No *EAC*-IV a operação de descartar um grupo $\mathbf{C}_s$ de uma partição é portanto realizada sobre a estrutura auxiliar, de forma que o grupo $\mathbf{C}_s$ é retirado na estrutura auxiliar (retirando a linha correspondente ao mesmo) e o grupo $\mathbf{C}_c$ mais próximo de $\mathbf{C}_s$, selecionado para receber os objetos desse último, tem a posição de seu centróide atualizada como sendo a média dos centróides de $\mathbf{C}_s$ e $\mathbf{C}_c$ ponderada pelo número de objetos de cada um desses grupos. No próximo passo do algoritmo evolutivo, a busca local, ao invés de fornecer o genótipo para o k-médias, fornece a estrutura auxiliar, superando desta forma a limitação de complexidade de tempo imposta pelo genótipo. Como o k-médias necessariamente irá avaliar cada um dos objetos do conjunto de dados para alocá-lo ao grupo mais adequado, é delegada apenas ao k-médias a atualização do genótipo.

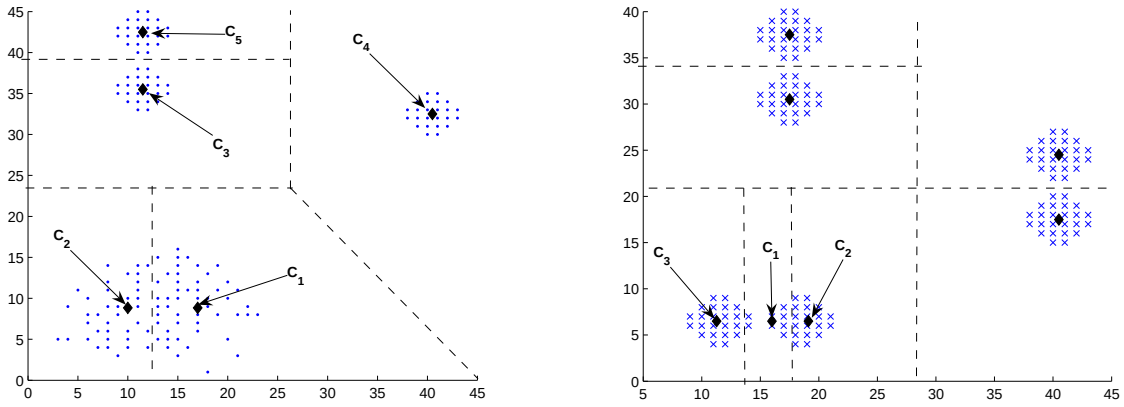
Claramente, deve existir um compromisso entre a eficiência computacional do operador proposto e a qualidade das partições obtidas.

Na Figura 3.11(a) é ilustrado um exemplo em que o *EAC*-IV será eficiente, enquanto na Figura 3.11(b) apresenta-se um caso em que a simplificação do OM_1 poderá resultar na geração de um

3	3	2	1	2	1	3	3	1	2
1	2	3	4	5	6	7	8	9	10

Figura 3.10: Genótipo $\mathbf{G}_1$.

genótipo de pior qualidade. Considerando que o grupo $\mathbf{C}_1$ (Figura 3.11(a)) seja selecionado para sofrer mutação, o agrupamento mais coerente resultante da remoção de $\mathbf{C}_1$ constitui-se na realocação de todos os seus objetos para o grupo $\mathbf{C}_2$, que é seu grupo mais próximo. No exemplo seguinte (Figura 3.11(b)) considere que o grupo $\mathbf{C}_1$ tenha sido selecionado para sofrer mutação. Embora o resultado esperado para o agrupamento seja a alocação dos seus objetos entre os grupos $\mathbf{C}_2$ e $\mathbf{C}_3$ (os dois objetos mais à esquerda do grupo $\mathbf{C}_1$ alocados ao grupo $\mathbf{C}_3$ e os demais objetos ao grupo $\mathbf{C}_2$) a partição resultante, como definido no *EAC*-IV, terá todos os objetos de $\mathbf{C}_1$ alocados ao grupo $\mathbf{C}_2$, que é o mais próximo de $\mathbf{C}_1$ segundo a distância dos centróides.



(a) (b)

Figura 3.11: Dois conjuntos de dados: (a) exemplo em que é adequado o *EAC*-IV; (b) exemplo onde o *EAC*-IV mostra-se inadequado.

3.2.5 *EAC*-V

Em alternativa à seleção original por roleta utilizada no *EAC* [Hruschka et al., 2006] utiliza-se aqui a seleção determinística $(\mu + \lambda)$ [Fogel, 1999], abordagem comumente adotada em estratégias evolutivas. Nessa abordagem, os μ melhores indivíduos da população na geração t serão pais de λ filhos, constituindo assim a população de $(\mu+\lambda)$ indivíduos na geração $t+1$. Apesar de teoricamente aumentar a susceptibilidade da busca a ótimos locais (pelo aumento da pressão seletiva em decorrência da seleção determinística), a opção da avaliação desta abordagem vem de uma tentativa prévia da adoção do método do torneio [Fogel, 1999] na esperança de que, ao diminuir a pressão seletiva, melhores resultados seriam obtidos. Como essa hipótese não se confirmou, optou-se então pela avaliação de uma estratégia determinística, que possui maior pressão seletiva. É possível observar aplicações de sucesso da seleção determinística a problemas de agrupamento de dados em [Fränti et al., 1997].

3.2.6 Análise de eficiência computacional dos algoritmos propostos

O custo total do *EAC* foi estimado como $O(N)$, onde N é o número de objetos do conjunto de dados [Hruschka et al., 2006]. Esse custo advém essencialmente do critério da Silhueta Simplificada (seção 3.3.2) e das operações que requerem percorrer todas as N posições do genótipo. Esta estimativa não é afetada por nenhuma das modificações propostas que resultaram nos algoritmos evolutivos para agrupamento de dados descritos neste trabalho. Entretanto, algumas dessas modificações resultam em ganhos significativos em termos de tempo computacional de processamento, como será reportado na seção 4.1. É importante observar que estratégias de amostragem [Kaufman & Rousseeuw, 1990] podem ser usadas para aplicar algoritmos de agrupamento de dados em grandes conjuntos de dados. Isso é importante uma vez que N pode ser suficientemente grande para tornar proibitivo mesmo o uso de um algoritmo com custo linear $O(N)$, caso a constante do custo não seja reduzida a valores tratáveis

3.3 Funções de adequabilidade

O *EAC* originalmente emprega a Silhueta Simplificada (que será detalhada na seção 3.3.2) como sua função de adequabilidade. Embora a Silhueta Simplificada tenha se mostrado útil em diversos conjuntos de dados (veja [Hruschka et al., 2004b, Hruschka et al., 2004c, Hruschka et al., 2006]), pode-se considerar que, para aplicações específicas, outras funções de adequabilidade podem ser mais indicadas. Nesse contexto, é importante destacar que o conceito de "número correto de grupos" é subjetivo, assim como o próprio conceito de "grupo", cuja qualidade é avaliada pelos critérios de validade de agrupamento (funções de adequabilidade). Logo, torna-se de particular interesse nesse trabalho a avaliação do desempenho dos algoritmos propostos de acordo com um critério absoluto para avaliar partições, tal como o *Rand Index* [Jain & Dubes, 1988]. Desta forma, a análise pode ser centralizada no desempenho dos operadores evolutivos desenvolvidos, porque a sensibilidade da função objetivo em relação às bases de dados empregadas é reduzida. Ao que tudo indica, tal abordagem não tem sido usada para avaliar algoritmos evolutivos para agrupamento de dados, e parece ser uma metodologia promissora que merecerá melhor atenção no futuro. A despeito da motivação "natural" do uso de um critério absoluto como o *Rand Index*, utiliza-se aqui também a Silhueta Simplificada do *EAC* original como função de adequabilidade, com o intuito de ilustrar a aplicação dos algoritmos propostos em um cenário mais prático.

3.3.1 *Rand Index*

O *Rand Index* [Rand, 1971] pode ser visto como um critério absoluto (externo) ou como um padrão referencial que permite o uso de um conjunto de dados de classificação para realizar avaliação não somente de classificadores (que podem produzir diferentes partições de dados com o número correto de classes), mas também de resultados de agrupamentos de dados (em que diferentes partições de dados podem ser compostas de diferentes números de grupos). Este índice simples e intuitivo avalia duas partições rígidas (*hard* ou *crisp*) **R** e **C** do mesmo conjunto de dados. A partição de referência, **R**, codifica o rótulo das classes, ou seja, ela particiona o conjunto de dados em k^* classes conhecidas. A partição **C**, por sua vez, secciona o conjunto de dados em k categorias (classes ou grupos) e é aquela a ser avaliada. As categorias codificadas por **C** serão, daqui em diante, chamadas de grupos, pelo contexto de algoritmos de agrupamento de dados do presente trabalho. Desta forma, é possível distinguir entre estas e as classes corretas codificadas por **R**. Então, após estas observações, o *Rand Index* é definido como:

$$\Omega(\mathbf{R}, \mathbf{C}) = \frac{a+d}{a+b+c+d} \tag{3.4}$$

Em que:

- a: Número de pares de objetos do conjunto que pertencem à mesma classe em **R** e ao mesmo grupo em **C**.
- b: Número de pares de objetos que pertencem à mesma classe em **R** e a diferentes grupos em **C**.
- c: Número de pares de objetos que pertencem a classes diferentes em **R** e ao mesmo grupo em **C**.
- d: Número de pares de objetos que pertencem a classes distintas em **R** e a grupos distintos em **C**.

Os termos a e d são medidas de classificações consistentes, enquanto os termos b e c são medidas de classificações inconsistentes. Note que: (i) $\Omega \in [0, 1]$; (ii) $\Omega = 0$ se e somente se **C** é completamente inconsistente, ou seja, $a = d = 0$; e (iii) $\Omega = 1$ se e somente se a partição sob avaliação corresponde exatamente à partição de referência, ou seja, $b = c = 0$ ($\mathbf{C} = \mathbf{R}$).

O *Rand Index* em (3.4) provê uma medida de qualidade da partição total. Os algoritmos evolutivos descritos nas Seções 3.2.1 e 3.2.2, entretanto, demandam uma medida adicional de qualidade para cada grupo separadamente (função de adequabilidade parcial). Por essa razão, é proposto aqui um

novo conceito, o *Rand Index* para um grupo. O *Rand Index* do i-ésimo grupo $\mathbf{C}_i$ ($i \in 1, \ldots, k$) da partição **C** é definido como:

$$\Omega_i(\mathbf{R}, \mathbf{C}) = \frac{a_i + d_i}{a_i + b_i + c_i + d_i} \tag{3.5}$$

Em que:

- a_i: Número de pares de objetos pertencentes à mesma classe em **R** e ao mesmo grupo $\mathbf{C}_i$ em **C**.
- b_i: Metade do número de pares de objetos pertencentes à mesma classe em **R** e a grupos distintos $\mathbf{C}_i$ e $\mathbf{C}_j$ em **C** (onde $j \in 1, \ldots, k$ e $j \neq i$).
- c_i: Número de pares de objetos que pertencem a classes distintas em **R** e ao mesmo grupo $\mathbf{C}_i$ em **C**.
- d_i: Metade do número de pares de objetos pertencentes a classes distintas em **R** e a grupos distintos $\mathbf{C}_i$ e $\mathbf{C}_j$ em **C** (onde $j \in 1, \ldots, k$ e $j \neq i$).

Note que somente metade do número de pares de objetos nos termos b_i e d_i é computada. A razão é que somente um dos objetos de cada par pertence ao grupo $\mathbf{C}_i$ sob avaliação. A outra metade será contada quando computado o *Rand Index* dos outros grupos $\mathbf{C}_j$ ($i \neq j$).

É possível demonstrar que o *Rand Index* da partição completa **C** na Eq. (3.4) é a média ponderada dos *Rand Indexes* individuais Ω_i na Eq. (3.5) ($i = 1, \ldots, k$), com o peso de Ω_i proporcional ao número de objetos no grupo $\mathbf{C}_i$. Tal demonstração encontra-se no Apêndice A.

3.3.2 Silhueta Simplificada

Antes de definir a Silhueta Simplificada, apresenta-se a seguir a Silhueta original proposta por [Kaufman & Rousseeuw, 1990]. Para tanto, considere um objeto $\mathbf{x}_i$ pertencente a um dado grupo $\mathbf{C}_a$. Então, a distância média de $\mathbf{x}_i$ para os demais objetos de $\mathbf{C}_a$ é denotada por $a(\mathbf{x}_i)$. Levando em conta um outro grupo $\mathbf{C}_c$, a distância média do objeto $\mathbf{x}_i$ para todos os objetos do grupo $\mathbf{C}_c$ será referenciada por $d(\mathbf{x}_i, \mathbf{C}_c)$. Após o cálculo de $d(\mathbf{x}_i, \mathbf{C}_c)$ para todos os grupos $\mathbf{C}_c \neq \mathbf{C}_a$, o menor valor é selecionado, ou seja, $b(\mathbf{x}_i) = min[d(\mathbf{x}_i, \mathbf{C}_c)]$, $\mathbf{C}_c \neq \mathbf{C}_a$. Este valor representa a distância de $\mathbf{x}_i$ para o grupo mais próximo, e a Silhueta $s(\mathbf{x}_i)$ é dada pela equação (3.6):

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{max[a(\mathbf{x}_i), b(\mathbf{x}_i)]} \tag{3.6}$$

É de fácil verificação que $s(\mathbf{x}_i) \in [-1, 1]$ e que, quanto maior o valor de $s(\mathbf{x}_i)$, melhor a alocação de $\mathbf{x}_i$ para um dado grupo. Adicionalmente, se $s(\mathbf{x}_i)$ é igual a zero, então não fica claro para qual grupo o objeto $\mathbf{x}_i$ deveria ser alocado, no seu grupo atual ou no seu grupo vizinho. Finalmente,

se o grupo $\mathbf{C}_a$ for unitário (*singleton*), então $s(\mathbf{x}_i)$ é não definida e a escolha mais neutra é atribuir $s(\mathbf{x}_i) = 0$ [Kaufman & Rousseeuw, 1990]. O critério da Silhueta é dado pela média de $s(\mathbf{x}_i)$ com $i = \{1, \ldots, N\}$. A melhor partição de dados segundo esse critério é atingida quando o mesmo é maximizado.

A Silhueta proposta por [Kaufman & Rousseeuw, 1990] e definida acima depende do cálculo de todas as distâncias entre os objetos da base, levando a um custo computacional de $O(N^2)$, que é proibitivo para a maioria das aplicações reais de agrupamento de dados. Para superar essa limitação, uma Silhueta Simplificada [Hruschka et al., 2004c, Hruschka et al., 2006] pode ser empregada. A Silhueta Simplificada é baseada no cálculo das distâncias entre objetos e os centróides dos grupos. Mais especificamente, o termo $a(\mathbf{x}_i)$ da equação (3.6) torna-se a distância do objeto $\mathbf{x}_i$ ao centróide do seu grupo ($\mathbf{C}_a$). Similarmente, no lugar do cálculo de $d(\mathbf{x}_i, \mathbf{C}_c)$ como a distância média do objeto $\mathbf{x}_i$ para todos os objetos de $\mathbf{C}_c$, $\mathbf{C}_c \neq \mathbf{C}_a$, somente a distância entre $\mathbf{x}_i$ e o centróide de $\mathbf{C}_c$ é calculada. Enquanto essas modificações reduzem o custo de $O(N^2)$ para $O(N)$, resultados empíricos [Hruschka et al., 2004c, Hruschka et al., 2006] indicam que a qualidade das partições encontradas parece não ser significantemente afetada.

A computação da Silhueta original, assim como a da sua versão simplificada, depende apenas da partição obtida e não do algoritmo de agrupamento de dados empregado. Portanto, essas Silhuetas podem ser aplicadas para avaliar partições (levando em conta o número de grupos) obtidas por diversos algoritmos de agrupamento de dados. Neste trabalho, a Silhueta Simplificada é empregada como a função de adequabilidade prática a ser utilizada em algoritmos evolutivos para agrupamento de dados. Sendo o algoritmo das k-médias usado como busca local, isso garante que $a(\mathbf{x}_i) \leq b(\mathbf{x}_i)$. Assim, tem-se que $s(\mathbf{x}_i) \in [0, 1]$. Finalmente, sendo a Silhueta Simplifica definida em termos de média de $s(\mathbf{x}_i)$ para todos os objetos da base, é possível calcular a função parcial associada a um dado grupo como a média das Silhuetas $s(\mathbf{x}_i)$ tomada apenas sobre os objetos que pertencem a esse grupo. Essa propriedade é especialmente útil para os algoritmos descritos nas Seções 3.2.1 e 3.2.2, que necessitam da função de adequabilidade parcial para os grupos.

3.4 Algoritmo *Fast-EAC*

Na seção 4.1.3 os cinco novos algoritmos propostos na seção 3.2 serão empiricamente avaliados. Em resumo, faz-se conveniente adiantar aqui que os algoritmos *EAC*-II, *EAC*-III e *EAC*-V apresentarão resultados significativamente melhores que o *EAC* original, podendo ser portanto combinados em uma nova ferramenta, o Algoritmo Evolutivo Rápido para Agrupamento de Dados (*Fast-EAC*).

O *Fast-EAC* será um novo algoritmo baseado no *EAC*, com as seguintes características: (i) seleção guiada de grupos para mutação (*EAC*-II); (ii) auto-ajuste das taxa de aplicação dos operadores

de mutação (*EAC*-III); (iii) Seleção determinística $(\mu + \lambda)$ (*EAC*-V). Assumindo que essas características apresentam comportamentos independentes entre si, espera-se que o algoritmo *Fast-EAC* seja superior a todas as novas propostas de melhorias do *EAC* apresentadas nesse trabalho quando consideradas isoladamente.

Os principais passos do *Fast-EAC* são apresentados no Algoritmo 6

Algoritmo 6 Pseudo-código para o Algoritmo Evolutivo Rápido para Agrupamento de Dados.

1: [Partições] = **algoritmo *Fast EAC***(**X**, T)
2: Gere a população inicial aleatoriamente;
3: **Enquanto** o critério de convergência não é satisfeito **faça**
4: Aplique o algoritmo das k-médias a todos os genótipos;
5: Avalie cada genótipo de acordo com a função de adequabilidade;
6: Selecione os genótipos para mutação pela seleção determinística $(\mu + \lambda)$;
7: Calcule a função de adequabilidade parcial de cada grupos de cada genótipo;
8: Calcule a probabilidade de mutação de cada grupo por meio da linearização dos valores de adequabilidade parcial calculados no passo anterior;
9: Calcule as taxas de sucesso dos operadores de mutação na geração anterior (equações 3.2 e 3.3);
10: Aplique os operadores de mutação proporcionalmente às suas taxa de sucesso e sobre os grupos selecionados de acordo com suas probabilidades de mutação;
11: Substitua os genótipos antigos pelos formados no Passo 10;
12: **Fim enquanto**

3.5 Resumo

Nesse capítulo foram elaboradas melhorias ao Algoritmo Evolutivo para Agrupamento de Dados (*EAC*) [Hruschka et al., 2004c, Hruschka et al., 2006], que é capaz de estimar automaticamente o número de grupos enquanto otimiza a partição correspondente. Cinco variações do *EAC* foram projetadas para melhorar a eficiência computacional do *EAC*. Na seção 4.1.3 apresentar-se-ão comparações entre o *EAC* original [Hruschka et al., 2006] e as versões propostas nesta dissertação. Tais comparações possibilitam concluir que três dos cinco algoritmos propostos são potencialmente capazes de melhorar significativamente a eficiência computacional do *EAC* original. Com base nestas comparações, um novo algoritmo evolutivo denominado *Fast-EAC* foi também proposto neste capítulo, sendo este constituído pela incorporação de todas as variações do *EAC* capazes de melhorar significativamente o desempenho do *EAC* original em termos de eficiência computacional.

Capítulo 4

Experimentos

Neste capítulo pretende-se avaliar empiricamente os algoritmos definidos na seção 3.2, assim como o novo *Fast-EAC* resultado da combinação dos novos algoritmos que apresentam resultados melhores em termos de tempo computacional em comparação com o *EAC* original.

Duas seções principais compõem este capítulo: a primeira apresenta a avaliação empírica dos cinco novos algoritmos propostos em relação ao *EAC* original. Ainda nessa seção, o *Fast-EAC* é avaliado em relação ao *EAC* original e em relação aos cinco novos algoritmos propostos. Na seção seguinte apresenta-se a análise de um caso onde o *Fast-EAC* é comparado a estratégias de busca baseadas em múltiplas execuções do algoritmo das k-médias.

4.1 Avaliação experimental

4.1.1 Conjunto de dados

A avaliação da exatidão de uma dada partição em termos de classificação usualmente requer conjuntos de dados para os quais os "grupos" (classes) são conhecidos *a priori*. Nos experimentos realizados, foram adotados sete conjuntos de dados em que as partições ideais são conhecidas. No conjunto de dados chamado 9-Gauss, que é apresentado na Figura 4.1, os grupos podem ser reconhecidos por inspeção visual. Mais especificamente, este conjunto de dados é formado por 900 objetos distribuídos em nove grupos balanceados, que foram aleatoriamente gerados de uma distribuição Gaussiana bidimensional com desvios padrão de 0,5 e centros (x,y) dados por (1;1), (1;3), (1;5), (3;1), (3;3), (3;5), (5;1), (5;3) e (5;5) [Hruschka et al., 2004a, Campello & Hruschka, 2006].

Foram também empregados cinco conjuntos de dados sintéticos de bioinformática, descritos em [Yeung et al., 2003], para testar os algoritmos evolutivos. Esses conjuntos de dados, aqui chamados Bio1, . . . ,Bio5, são formados por $N = 400$ objetos – com $\rho = 20$ atributos cada – particionados em

seis grupos aproximadamente balanceados. Adicionalmente, o conjunto de dados real Yeast Galactose [Yeung et al., 2001], no qual os objetos também são compostos por 20 atributos, foi selecionado para os experimentos. Assim como em [Yeung et al., 2001], um subconjunto de N=205 objetos onde os padrões de expressão refletem quatro categorias (grupos) foi selecionado no caso dessa última base.

4.1.2 Metodologia

Muitos testes foram realizados para avaliar a qualidade das partições obtidas por cada um dos algoritmos evolutivos. Os grupos conhecidos para os conjuntos de dados empregados serão chamados de classes daqui em diante, enquanto o termo grupo será referente aos conjuntos de objetos similares encontrados pelos algoritmos de agrupamento. Isso vem de encontro com a terminologia adotada para o *Rand Index* (seção 3.3.1), que é uma das medidas adotadas nesse trabalho para avaliar a qualidade de uma dada partição. É importante mencionar que, embora as classes correspondentes para cada objeto sejam conhecidas *a priori*, essa informação não é utilizada durante o processo de agrupar os dados quando o critério da silhueta é utilizado.

A tabela 4.1 provê valores típicos de *Rand Index* e Silhueta Simplificada obtidas em nossos experimentos para cada conjunto de dados. O primeiro critério é especialmente usual quando a avaliação é centrada na avaliação do desempenho dos algoritmos evolutivos, enquanto o segundo possibilita a simulação de aplicações práticas, em que a partição ideal não é conhecida. A tabela 4.1 mostra que partições de boa qualidade foram encontradas (o valor relatado para o *Rand Index* é de particular interesse para avaliar a exatidão das partições encontradas - veja seção 3.3.1). Todos os algoritmos evolutivos descritos neste trabalho são capazes de encontrar os valores reportados na Tabela 4.1, mas com diferentes tempos computacionais. Conseqüentemente, tais valores serão usados como referência para avaliar a eficiência computacional de cada algoritmo. Embora melhores partições (em termos de *Rand Index* e Silhueta Simplificada) poderiam ser obtidas pelos algoritmos propostos quando aplicados a estes conjuntos de dados, na prática a experiência sugere que a probabilidade que isso aconteça é baixa. Mesmo assumindo que melhores partições sejam atingidas, o uso dos valores referência apresentados na tabela 4.1 continuam válidos, já que todos os algoritmos procuram por partições de qualidade equivalente.

Em termos de números de grupos, nos conjuntos de dados Bio1, Bio4, Bio5 e 9-Gauss foi possível encontrar grupos que correspondem precisamente às classes das partições de referência. No caso específico do conjunto de dados 9-Gauss, em razão da sobreposição das classes não foi possível encontrar partições exatamente iguais à partição de referência, apesar do número correto de grupos. Nos experimentos utilizando a Silhueta Simplificada como função de adequabilidade, particularmente nos conjuntos de dados Bio2, Bio3 e Yeast, não foi possível encontrar o número de grupos tais como

Tabela 4.1: Valores de referência para avaliação de partições (k_{RI} e k_{SS} indicam valores típicos de número de grupos encontrados pelo *Rand Index* e Silhueta Simplificada, respectivamente).

Conjunto de dados	*Rand Index*	k_{RI}	Silhueta Simplificada	k_{SS}
Bio1	1.000	6	0.775	6
Bio2	0.976	8	0.799	5
Bio3	0.991	7	0.807	4
Bio4	1.000	6	0.817	6
Bio5	1.000	6	0.766	6
Yeast	0.990	4	0.684	3
9-Gauss	0.974	9	0.609	9

as classes nas respectivas partições de referência. Isso porque, para a função de adequabilidade utilizada, algumas classes são tão parecidas entre si que faz mais sentido agrupá-las como um único grupo. Especificamente, no conjunto Bio2, em que foram encontrados cinco grupos, a partição obtida corresponde a quatro grupos corretos e um grupo dado pela união das classes rotuladas como “1” e “2” na partição referência. No conjunto Bio3, no qual foram encontrados quatro grupos, a partição obtida corresponde a dois grupos corretos, um grupo formado pela como a união das classes rotuladas como “1” e “2” e outro grupo dado pela união das classes rotuladas como “3” e “4” na partição de referência. No conjunto de dados Yeast, obtém-se 2 grupos formados corretamente e um terceiro grupo como a união das classes rotuladas como “2” e “3” na partição de referência.

Nos experimentos utilizando o *Rand Index* como função de adequabilidade, particularmente no que se refere aos conjuntos de dados Bio2 e Bio3, não foi possível encontrar o número correto de grupos. Na base Bio2 foram encontrados seis grupos correspondentes às classes da partição de referência, mas foram separados dois *outliers*, enquanto na base Bio3 foram encontrados seis grupos correspondentes às classes da partição padrão e foi separado um objeto como *outlier*.

Populações formadas por 4 e 20 genótipos foram utilizadas nos experimentos. Essa escolha pode ser considerada arbitrária; entretanto, o tamanho da população e o número de gerações requerido para convergência são tipicamente relacionados. De um ponto de vista prático, pode-se esperar que mais genótipos necessitem de menos gerações para atingir a convergência. Para a seleção determinística $(\mu+\lambda)$ adotou-se $\mu = 2$ e $\lambda = 2$ para populações com 4 indivíduos e $\mu = 4$ e $\lambda = 16$ para populações com 20 indivíduos. Na população formada por 4 indivíduos, optou-se pelos valores acima citados porque estes são os únicos que garantem equilíbrio entre os operadores evolutivos, outros valores possíveis de λ, i.e. $\lambda \in \{1, \ldots, 3\}$, implicariam que um dos dois operadores evolutivos do *EAC* seria mais aplicado que o outro. Na população de 20 indivíduos optou-se simplesmente por uma proporção diferente daquela utilizada na população de tamanho T=20, no intuito de verificar a robustez do algoritmo a este parâmetro. Em termos práticos, a escolha dos valores de μ e λ é arbitrária.

A população inicial é gerada de forma tal que o número de grupos k é aleatoriamente escolhido do conjunto $\left\{2, \ldots, N^{1/2}\right\}$, que é um critério heurístico usual para escolher valores aceitáveis para k [Pal & Bezdek, 1995]. Essa abordagem favorece a diversidade na população inicial, criando partições iniciais que representam diferentes números de grupos.

Os conjuntos de dados foram normalizado no intervalo [0,1], e a busca local (k-médias) foi programada para parar quando um dos seguintes critérios é satisfeito: (i) 5 iterações completas - evidências empíricas sugerem que cinco ou menos iterações do k-médias são suficientes [Anderberg, 1973]; ou (ii) a diferença máxima absoluta entre centróides em duas iterações consecutivas é menor ou igual a 0.001.

Foram executados 50 experimentos para cada possível cenário resultante da combinação das seguintes configurações: 6 algoritmos (*EAC*, *EAC*-I, *EAC*-II, *EAC*-III, *EAC*-IV, *EAC*-V), 2 funções de adequabilidade (*Rand Index* e Silhueta Simplificada), 7 conjuntos de dados (9-Gauss, Bio1, ..., Bio5, Yeast)e 2 tamanhos de população ($T = 4$ e $T = 20$ genótipos). Em resumo, 8.400 $(50 \times 6 \times 2 \times 7 \times 2)$ experimentos foram realizados inicialmente para avaliar a eficiência dos algoritmos propostos. Todos os experimentos foram realizados em um Pentium IV, 3 GHz CPU, 1 ou 2 GB RAM, rodando apenas o sistema operacional e o ambiente do Matlab (usado em todas as simulações), garantindo que todos os algoritmos sejam executados na mesma estação de processamento para uma mesma base, até um dos seguintes critérios serem atingidos: (i) os valores meta para função adequabilidade (Tabela 4.1) ou maiores serem encontrados; ou (ii) 500 gerações serem concluídas. Especificamente, quando os algoritmos alcançam um dos critérios, seu tempo computacional é armazenado. Observou-se que o número de gerações é em geral correlacionado com o tempo de CPU para esses experimentos. Entretanto, o mesmo número de gerações, em diferentes experimentos, podem conduzir a diferentes tempos de CPU, em razão da natureza probabilística da busca evolutiva. Conseqüentemente, nesses experimentos o tempo computacional é adotado como critério de comparação.

4.1.3 Resultados e análise experimental

As tabelas 4.2, 4.3, 4.4 e 4.5 relatam o tempo computacional (em segundos) requerido pelos algoritmos para alcançar os valores de referência de *Rand Index* e Silhueta Simplificada reportadas na tabela 4.1. Pode-se observar pelas tabelas que o *EAC*-I, *EAC*-II, *EAC*-III, *EAC*-IV e *EAC*-V mostraram melhores resultados que o *EAC* na maioria dos conjuntos de dados. Entretanto, é mais interessante apresentar a comparação de desempenho em termos de eficiência total dos algoritmos, de forma que cada par de algoritmos (*EAC* original contra cada um dos algoritmos propostos) possa ser avaliado considerando todos os cenários de simulação juntos. Com esse propósito, os resultados foram avaliados estatisticamente por meio de análise de variância de dois critérios (ANOVA) [Johnson & Wichern, 1998]. A adoção desta análise favorece conclusões mais abrangentes, por con-

siderar as diferentes bases e tamanhos de população conjuntamente. Desta forma, pode-se verificar até onde os resultados entre os diversos algoritmos têm diferenças estatísticas em razão das modificações propostas (primeiro critério), o quanto destas diferenças estão relacionadas à escolha dos conjuntos de dados e do tamanho de população (segundo critério). Em outras palavras, ao adotar a análise de variância, busca-se verificar se a variância encontrada entre diferentes tratamentos (algoritmos) é maior que a variância encontrada entre os diferentes blocos (conjuntos de dados e população). Uma revisão geral dos métodos ANOVA pode ser encontrada em [Johnson & Wichern, 1998, Triola, 1999]. Assumiu-se a hipótese de normalidade dos resultados dos experimentos com base no teorema central do limite [Triola, 1999] e utilizou-se a transformação log para aproximar as variâncias entre as amostras.

Tanto para *Rand Index* quanto para Silhueta Simplificada, os resultados suportam que há desempenho significativamente diferente favorecendo o *EAC*-II, o *EAC*-III e o *EAC*-V com nível de significância $\alpha = 5\%$ quando comparados com o *EAC*. O *EAC*-I, diferentemente da sua versão com adequabilidade parcial linearizada (*EAC*-II), superou o *EAC* com $\alpha = 5\%$ somente nos experimentos com *RandIndex*. Sobre o *EAC*-IV, como outrora observado, a economia de tempo computacional do seu operador de mutação modificado OM_1 é contrabalanceada por um número maior de gerações necessárias para convergência. É presumido que isso aconteça em razão da perda de informação resultante da simplificação do operador (veja seção 3.2.4).

Em resumo, os resultados suportam que o *EAC*-II, *EAC*-III e *EAC*-V provêem ganhos de eficiência com significância estatística quando comparados ao *EAC*. Baseado nestes resultados, foi investigado o desempenho de um algoritmo combinando as características do *EAC*-II, *EAC*-III e *EAC*-V em uma só variação do *EAC*. Tal algoritmo, o *Fast-EAC* apresentado na seção 3.4, representa a contribuição central da presente dissertação. Possivelmente, uma combinação diferente entre os algoritmos propostos possa resultar em um algoritmo mais eficiente que o *Fast-EAC*. No entanto, como uma bateria de experimentos completa pode despender mais de trinta horas de processamento e os cinco algoritmos propostos podem ser combinados em mais de quarenta formas distintas, optou-se por avaliar somente a combinação resultante dos algoritmos que superaram o *EAC* com diferença estatística significativa.

O *Fast-EAC* constitui então um novo algoritmo baseado no *EAC*, com as seguintes características: (i) seleção guiada de grupos para mutação (*EAC*-II); (ii) auto-ajuste das taxas de aplicação dos operadores de mutação (*EAC*-III); e (iii) seleção determinística $(\mu + \lambda)$ (*EAC*-V).

Com as mesmas condições dos experimentos anteriores, foi realizado um conjunto de 1.400 experimentos adicionais para avaliar o *Fast-EAC* (50 experimentos por conjunto de dados $\times$ 7 conjuntos de dados $\times$ 2 funções de adequabilidade $\times$ 2 tamanhos de população). Os resultados principais foram resumidos nas últimas linhas das tabelas 4.2 até 4.5.

Novamente, ANOVA foi usado para avaliar o desempenho do algoritmo combinado (*Fast-EAC*),

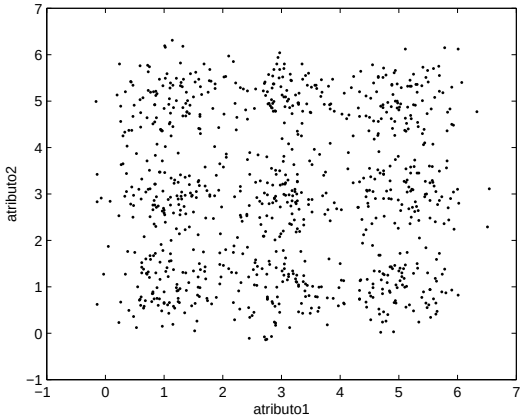


Figura 4.1: Conjunto 9-Gauss.

Tabela 4.2: Tempos Computacionais (*Rand Index* - 4 Genótipos).

	Bio1	**Bio2**	**Bio3**	**Bio4**	**Bio5**	**Yeast**	**9-Gauss**
EAC	15,3	236	226,7	10,3	15,6	5,9	555,7
***EAC*-I**	7,4	89,6	72,3	7,5	7,3	4,6	638,3
***EAC*-II**	13,2	185,2	43	10,1	14,6	4,5	639,7
***EAC*-III**	10	167,9	119	7,4	12,8	4,3	193,2
***EAC*-IV**	13,6	129,5	167,1	17	14,4	5,7	451,8
***EAC*-V**	13	187,7	46,6	12	15,5	34	392,9
Fast-EAC	8,6	146,4	41,4	6,8	10,7	29,3	361,3

Tabela 4.3: Tempos Computacionais (*Rand Index* - 20 Genótipos).

	Bio1	**Bio2**	**Bio3**	**Bio4**	**Bio5**	**Yeast**	**9-Gauss**
EAC	28	275,7	344,5	21,2	24,1	12,1	482,7
***EAC*-I**	22,7	107,9	141,5	19,6	23,6	12,3	537,5
***EAC*-II**	24,9	214,9	102,5	19,4	27,4	12,5	416,7
***EAC*-III**	25,4	254,9	264,3	20,8	25,8	12,4	329
***EAC*-IV**	22	226	442,5	20,8	23,6	12	473,7
***EAC*-V**	22,1	125,8	159,9	20,7	22,1	61,7	270,8
Fast-EAC	23,6	140,8	50,1	21,6	22,1	11,2	195,8

que se mostrou mais eficiente que todos os seus predecessores com nível de significância $\alpha = 5\%$. Desta forma, esta nova ferramenta se mostra promissora para problemas de agrupamento de dados.

4.2 Experimento adicional

Adicionalmente aos experimentos apresentados na seção 4.1, será apresentado nessa seção uma comparação entre o *Fast-EAC* e algumas metodologias bem estabelecidas. Em [Hruschka et al., 2006], o *EAC* mostrou-se, por meio de análises experimentais, ser mais rápido em comparação a uma metodologia de múltiplas execuções do algoritmo das k-médias. Essa seção tem como objetivo ilustrar resultado análogo para o *Fast-EAC*. Na seção anterior, tendo o *Fast-EAC* se mostrado mais eficiente que o *EAC* com diferença estatisticamente significativa em todos os conjuntos de dados utilizados em [Hruschka et al., 2006], será utilizado agora outro conjunto de dados sintético oriundo de um gerador de bases específico para agrupamento de dados descrito em [Handl & Knowles, 2005]. Tal conjunto é formado por 1164 objetos com 10 dimensões e 20 grupos desbalanceados com distribuição Gaussiana multivariada.

4.2.1 Metodologia

Duas estratégias são utilizadas para comparar o desempenho em termos de tempo computacional do *Fast-EAC*: a primeira (busca seqüencial) consiste na execução do k-médias utilizando diversas inicializações aleatórias dos centróides iniciais para cada número de grupos dentro de um determinado intervalo. A segunda (busca por amostragem) consiste na execução do k-médias sorteando aleatoriamente tanto o número de grupos em um intervalo quanto os centróides iniciais.

A busca seqüencial foi utilizada para gerar os padrões (resultados referenciais) de comparação, de forma tal que essa busca é executada como primeiro passo da avaliação, gerando as soluções que servirão de meta para os demais algoritmos. É esperado que a busca evolutiva mostre-se capaz de

Tabela 4.4: Tempos Computacionais (Silhueta Simplificada - 4 Genótipos).

	Bio 1	**Bio 2**	**Bio 3**	**Bio 4**	**Bio 5**	**Yeast**	**9-Gauss**
EAC	16,17	21,71	5,42	16,21	30,37	3,55	122,31
***EAC*-I**	11,84	24,03	6,05	10,25	20,31	3,73	491,07
***EAC*-II**	14,42	9,51	6,89	11,31	20,18	3,87	116,16
***EAC*-III**	14	20,4	5,81	10,14	17,76	3,79	98,19
***EAC*-IV**	41,69	21,57	5,45	26,65	64,82	3,79	97,85
***EAC*-V**	20,69	10,19	5,71	16,06	18,63	3,33	107,01
Fast-EAC	12,64	8,32	6,91	12,97	21,83	3,12	83,1

encontrar soluções de qualidade similar em tempo computacional igual ou menor do que as buscas seqüencial e por amostragem.

Busca seqüencial

A busca seqüencial trata-se de uma estratégia bastante simples. Dado um conjunto de dados **X** com N objetos, será executado n_p vezes o algoritmo das k-médias para cada número de grupos, variando o número de grupos k no intervalo $k \in [2, \sqrt{N}]$. Conforme já mencionado anteriormente, o limitante superior desse intervalo tem origem heurística [Pal & Bezdek, 1995]. Para cada execução do k-médias são sorteados os centróides iniciais da partição. Após a execução da busca seqüencial, a melhor partição é eleita a partir de algum critério numérico de validade de agrupamento (e.g. Silhueta), doravante denominado "função objetivo".

Em última análise, a busca seqüencial é capaz de encontrar qualquer partição que possa ser encontrada pelo k-médias, bastando para isso um número suficientemente grande para n_p. Esse parâmetro é crítico para a busca, uma vez que ele garante a diversidade de inicializações para cada número de grupos k necessária para minimizar os efeitos da susceptibilidade do k-médias a ótimos locais. Dessa forma, a adoção de valores para n_p subestimados poderá comprometer a confiança sobre a qualidade do agrupamento, enquanto a adoção de valores de n_p superestimados poderá comprometer a eficiência computacional da estratégia.

Busca por amostragem

A busca por amostragem de grupos, por sua vez, representa uma busca estocástica no que tange tanto à seleção dos centróides iniciais como à seleção do número de grupos. Dado um conjunto de dados **X** com N objetos e um valor referencial de função objetivo β, é executado o k-médias tantas vezes forem necessárias para se atingir o valor meta β da função objetivo, sorteando, com distribuição uniforme e com reposição, o número de grupos k, com $k \in [2, \sqrt{N}]$. O método de inicialização dos

Tabela 4.5: Tempos Computacionais (Silhueta Simplificada - 20 Genótipos).

	Bio 1	**Bio 2**	**Bio 3**	**Bio 4**	**Bio 5**	**Yeast**	**9-Gauss**
EAC	29,05	60,76	20,77	23,14	32,59	12,42	181,19
***EAC*-I**	26,42	55,43	21,27	22,41	28,11	12,09	203
***EAC*-II**	30,63	22,06	22,08	24,12	26,21	12,83	189,93
***EAC*-III**	29,18	59,6	21,07	22,03	25,13	11,42	176,66
***EAC*-IV**	40,54	58,76	20,54	25,22	62,86	12,31	206,06
***EAC*-V**	25,04	58,71	20,6	22,29	26,63	10,25	136,28
Fast-EAC	25,23	22,05	22,78	22,72	25,6	10,35	163,98

centróides segue a mesma metodologia da busca seqüencial, que é aquela mesma adotada na geração da população inicial do *EAC*, descrita na seção 3.1.3.

Em razão do algoritmo das k-médias ser susceptível a ótimos locais, utiliza-se o sorteio com reposição do número k de grupos. De fato, considere que um determinado valor $k = k'$ de número de grupos corresponde ao único valor para o qual é possível obter uma partição com função objetivo maior ou igual ao valor meta. Se tal valor k' já foi amostrado no processo de busca e a partição obtida pelo k-médias a partir dos k' protótipos iniciais sorteados resultarem em uma partição com função objetivo inferior à meta, somente será possível atingir a meta se o sorteio do número k de grupos for com reposição.

Como k possui distribuição teórica de probabilidade uniforme, conforme o aumento do número de amostragens mais a distribuição prática de probabilidade se assemelha a uma distribuição uniforme. Conseqüentemente, para instâncias do problema com espaço de busca complexo, nos quais é necessário um número "elevado" de amostragens dos protótipos iniciais para que se atinja o valor meta da função objetivo, os resultados da busca por amostragem irão se aproximar dos resultados da busca seqüencial, uma vez que a distribuição do número k de grupos na busca por amostragem irá se assemelhar à distribuição determinística do número k de grupos da busca seqüencial. Tal propriedade confere uma relação bidirecional de que qualquer partição encontrada pela busca seqüencial pode ser encontrada na busca por amostragem, dependendo exclusivamente do ajuste de parâmetros de número de execuções para a busca por amostragem (relacionado ao valor meta da função objetivo) e do número de partições por grupo n_p para a busca seqüencial.

Índice de Jaccard como opção ao *Rand Index*

Diferente da avaliação experimental realizada na seção 4.1, será adotado aqui o índice de Jaccard [Jain & Dubes, 1988] para avaliar a qualidade das partições segundo o contexto de classificação. A opção por avaliar as partições utilizando o índice de Jaccard ao invés do *Rand Index* deve-se à tendência desse último, em alguns cenários particulares, de apresentar valores altos em razão da quantidade de grupos e classes nas partições. Especificamente, partições com números superestimados de grupos tendem a apresentar valores para *Rand Index* elevados, especialmente quando o número de classes na partição de referência é elevado. Essa tendência deve-se ao termo d da equação (3.4), que pode dominar os demais termos no cálculo envolvendo números elevados de grupos e classes. Esse não é o caso das bases de dados utilizadas na seção 4.1, mas é o caso da base utilizada no presente experimento adicional.

A fim de ilustrar a influência do termo d sobre o cálculo do *Rand Index*, considere um conjunto de N objetos, para o qual se obtém uma partição com N grupos unitários (*singletons*). Se esta partição for avaliada pelo *Rand Index*, independente da partição de referência (padrão), os termos a e c serão

nulos e todos os pares de objetos serão classificados entre os termos b e d. Caso a partição padrão possua apenas duas classes de igual quantidade de objetos ($\frac{N}{2}$), então cada objeto se combina com $\frac{N}{2}$ objetos em pares classificados em d e com $\frac{N}{2} - 1$ objetos em pares classificados como b. Aumentando a quantidade de classes balanceadas na partição padrão claramente aumenta-se a concentração de d sobre b, de forma que $\Omega = \frac{d}{b+d} \rightarrow 1$.

Como exemplo, caso um conjunto de dados de 1998 objetos igualmente distribuídos em 18 classes na partição padrão seja comparado com uma partição contendo 1998 subconjuntos unitários (*singletons*), obtém-se um *Rand Index* $\Omega = 0.94$. Em resumo, conforme o aumento do número de grupos da partição padrão, maior a tolerância do *Rand Index* a classificações incorretas, visto que o número de pares de objetos classificados como d tenderá a crescer em relação aos demais termos da equação.

Em alternativa à utilização do *Rand Index* como critério externo de avaliação de partições pode-se adotar o índice de Jaccard, definido como [Jain & Dubes, 1988]:

$$J = \frac{a}{a + b + c} \tag{4.1}$$

Esse índice possui as mesmas propriedades de avaliação do *Rand Index*, sem a tendência de apresentar valores maiores em função do número de classes da partição padrão ou grupos na partição a ser avaliada. Particularmente, para avaliação das partições do conjunto de dados de 20 Gaussianas, a utilização do índice de Jaccard será mais adequado. Nos experimentos anteriores (seção 4.1), como as partições padrão possuem número bastante inferior de classes tem-se que a utilização do *Rand Index* não implica em uma avaliação inadequada.

4.2.2 Experimentos e análise de resultados

Com o propósito de avaliar o algoritmo evolutivo em contraposição às duas metodologias descritas na seção 4.2.1, foram executados 50 experimentos com a busca seqüencial. Ao final das execuções, avaliaram-se todas as partições geradas para cada um dos experimentos e selecionou-se a melhor partição, utilizando a Silhueta Simplificada como função objetivo. Tomou-se, então, o valor de Silhueta dessa partição como sendo o valor meta para os demais algoritmos (*Fast-EAC* e busca por amostragem). Como parâmetro para a busca seqüencial, utilizou-se $n_p = 10$ e $k \in [2, \sqrt{N}]$, executando o k-médias até um limiar mínimo de 10^{-3} para a diferença máxima entre a matriz de centróides em duas iterações consecutivas ou até um máximo de 100 iterações.

Na Tabela 4.6 são apresentadas médias e desvios padrão para tempo computacional (em segundos), índice de Jaccard, número de grupos e Silhueta Simplificada. Conforme explicado na seção 4.2.1 - "Busca seqüencial", o parâmetro n_p é crítico. No entanto, ao observar a variância tanto para valores de função objetivo quanto para número de grupos, pode-se concluir que tal parâmetro não

foi superestimado, uma vez que os experimentos não convergiram para partições equivalentes. Além disso, tem-se que a busca seqüencial é utilizada para estabelecer o valor meta da função objetivo, evitando problemas com a determinação do valor mínimo para n_p que permita a essa busca atingir algum limiar pré-estabelecido da função objetivo.

Tabela 4.6: Resultados da busca seqüencial: média (μ) e desvio padrão (σ) dos tempos computacionais.

	k **grupos**	**Jaccard**	**Silhueta**	**Tempo (s)**
μ	19,78	0,81	0,60	2.277,63
σ	3,20	0,08	0,01	101,24

A Tabela 4.7 apresenta um sumário de tempo de processamento, em segundos, para as três estratégias, isto é, busca seqüencial, busca por amostragem e *Fast-EAC*.

Tabela 4.7: Sumário de tempo de processamento (segundos): média (μ) e desvio padrão (σ).

	Seqüencial	**Amostragem**	***Fast-EAC***
μ	2.277,6	5.701,1	620,9
σ	101,2	7.567,4	544,3

Uma vez assumida a hipótese de normalidade dos resultados dos experimentos com base no teorema central do limite [Triola, 1999], pode-se aplicar o teste-t de Student [Triola, 1999] para verificar a hipótese de igualdade entre as médias de duas populações de resultados dos algoritmos. Com $\alpha = 5\%$ é possível concluir que o algoritmo evolutivo *Fast-EAC* é mais rápido, com significância estatística, que ambas as buscas seqüencial e por amostragem. Em termos de eficácia dos algoritmos é importante mencionar que partições de qualidade superior poderiam ser encontradas caso o critério de convergência pelo valor meta de função de adequabilidade não fosse utilizado. Nesse contexto, realizaram-se 5 experimentos adicionais, a título de ilustração, utilizando como critério de parada 100 gerações concluídas para o algoritmo evolutivo. Tais experimentos resultaram em partições com Silhueta Simplificada média $s = 0,628$, com índice de Jaccard de $J = 0,862$ e 18,6 grupos em média. A melhor partição encontrada ($s = 0,634$) possui índice de Jaccard $J = 0.884$ com 19 grupos, tais grupos correspondem a 18 das 20 classes conhecidas *a priori* separadas corretamente e duas classes agrupadas em um mesmo grupo.

4.3 Resumo

Os algoritmos propostos na seção 3.2 foram avaliados no presente capítulo em sete conjuntos de dados, usando duas funções de adequabilidade. Análises estatísticas de variância (ANOVA) foram realizadas possibilitando concluir que três variações do *EAC* mostraram-se mais eficientes que o *EAC* original, com relevância estatística. Tais variações foram incorporadas ao *EAC*, gerando um algoritmo evolutivo mais eficiente para agrupamento de dados, chamado Algoritmo Evolutivo Rápido para Agrupamento de Dados (*F-EAC*). Este algoritmo se mostra uma ferramenta promissora para problemas de agrupamento de dados. Foi descrita também uma nova metodologia de avaliação de algoritmos evolutivos para agrupamento de dados de forma que a influência da função de adequabilidade é reduzida no processo de avaliação, assim direcionando a análise especificamente nos operadores evolutivos. Foi apresentado um exemplo de comparação entre o *F-EAC* e estratégias de busca baseadas em múltiplas execuções do k-médias, em que o *Fast-EAC* mostrou-se mais rápido com diferença estatisticamente significativa.

Capítulo 5

Conclusões

Neste trabalho foi proposto um Algoritmo Evolutivo Rápido para Agrupamento de Dados (*Fast-EAC*). Tal algoritmo caracteriza-se por determinar automaticamente a estrutura e o número k de grupos que melhor representam um conjunto de dados segundo algum critério de validade de agrupamento para medir qualidade de partições. Ao utilizar o algoritmo das k-médias como procedimento de busca local, o *Fast-EAC* tende a buscar estruturas com distribuição geométrica induzida pela métrica daquele algoritmo; mais especificamente por grupos hiperesféricos quando a métrica Euclidiana é adotada.

Em termos de aplicabilidade, o *Fast-EAC* é, por construção, apropriado para aplicações em que se possa fazer uso do algoritmo das k-médias para o agrupamento, uma vez que o mesmo busca por partições da mesma natureza que o k-médias pode encontrar, independente dos centróides iniciais e do número k de grupos. Ainda em termos de aplicabilidade, em princípio o *Fast-EAC* é restrito a aplicações onde a medida de distância Euclidiana mostra-se adequada, uma vez que a mesma é utilizada para mensurar a dissimilaridade entre os objetos na função de adequabilidade e o algoritmo de busca local. No entanto, o *Fast-EAC* pode ser equipado com outra medida de dissimilaridade e outro mecanismo de busca local, com opções que sejam mais adequadas a um problema específico. Investigações nesse contexto fazem parte das perspectivas de trabalhos futuros da presente dissertação.

O *Fast-EAC* mostrou-se mais rápido, em sete conjuntos de dados e com significância estatística, do que o seu predecessor *EAC* [Hruschka et al., 2004c], que foi mostrado em [Hruschka et al., 2006] ser significativamente mais rápido que múltiplas execuções do k-médias em experimentos envolvendo 6 das 7 bases de dados utilizadas na Seção 4.1. Mostrou-se também mais rápido que a técnica de múltiplas execuções do k-médias em um conjunto de dados adicional.

Em resumo, as principais contribuições do presente trabalho são: (i) a criação de um novo algoritmo evolutivo para agrupamento de dados, o *Fast-EAC*; (ii) a proposição da metodologia de avaliação de algoritmos evolutivos para agrupamento utilizando critérios externos de avaliação, com o

objetivo de minimizar a influência da escolha da função objetivo sobre o desempenho dos algoritmos, buscando assim concentrar a análise nos operadores evolutivos; (iii) a proposição do *Rand Index* parcial, um novo índice externo de avaliação baseado no *Rand Index* [Rand, 1971] com capacidade de avaliação individual por grupos (*Rand Index* parcial), e sua relação com o *Rand Index* original de uma partição completa. Um aspecto importante sobre o *Rand Index* parcial é que ele possibilita análises mais sofisticadas sobre as partições de dados avaliadas (e, por conseqüência, sobre o algoritmo gerador). Ao gerar índices por grupos, pode-se avaliar, por exemplo, se uma partição possui grupos de qualidade similar ou se alguns dos grupos se destacam por terem qualidade acima ou abaixo da média. Outras análises envolvendo número de grupos, número de objetos por grupo e índice por grupos podem ser realizadas, abrindo meios para análises por diferentes perspectivas e que podem ser mais informativas no quesito eficácia do algoritmo em termos de classificação.

As principais contribuições dessa dissertação foram publicadas em [Alves et al., 2006]. Cumpre, ainda, ressaltar que, além das contribuições apresentadas neste documento, outros trabalhos vinculados à pesquisa desenvolvida foram publicados em [Alves et al., 2007, **?**]. Nestes, são propostas e avaliadas extensões do *EAC* para tratar com grupos de dados sobrepostos (*fuzzy clusters*).

Trabalhos futuros vislumbrados a partir deste incluem: (i) a extensão do *Fast-EAC* para a otimização de partições de dados com sobreposição de grupos, permitindo o tratamento de problemas em que a fronteira dos grupos não é bem delineada *Fast-EAC fuzzy*; (ii) a avaliação empírica de diferentes valores de μ e λ da seleção determinística, com o intuito de verificar mais detalhadamente a influência de tais parâmetros sobre o desempenho geral do algoritmo; (iii) o estudo de um mecanismo de mutação dinâmica para ajustar o escopo da mutação (número de grupos passíveis de mutação) de forma proporcional ao estágio evolutivo da população; (iv) a avaliação do desempenho do *Fast-EAC* com diferentes medidas de distância (e.g. Hiperelipsoidais Adaptativas), funções objetivo e mecanismos de busca local; (v) aplicações em problemas reais, como em segmentação de imagens e mineração de textos.

Referências Bibliográficas

[Alves et al., 2006] Alves, V. S.; Campello, R. J. G. B.; Hruschka, E. R. Towards a fast evolutionary algorithm for clustering. Em *IEEE International Conference on Evolutionary Computation*, pp. 6240–6247, 2006.

[Alves et al., 2007] Alves, V. S.; Campello, R. J. G. B.; Hruschka, E. R. A fuzzy variant of an evolutionary algorithm for clustering. Em *IEEE International Conference on Fuzzy Systems*, 2007.

[Anderberg, 1973] Anderberg, M. R. *Cluster Analysis for Applications*. Academic Press Inc., 1973.

[Bandyopadhyay & Maulik, 2002] Bandyopadhyay, S.; Maulik, U. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition* **35**(6), pp. 1197–1208, 2002.

[Bäck et al., 2000] Bäck, T.; Fogel, D. B.; Michalewicz, Z. (eds). *Evolutionary Computation 1: Basic Algorithms and Operators*. IOP Publishing Ltd., Bristol, UK, UK, 2000.

[Bäck et al., 1997] Bäck, T.; Hammel, U.; Schwefel, H.-P. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutyonary Computation* **1**(1), 1997.

[Bobrowski & Bezdek, 1991] Bobrowski, L.; Bezdek, J. C. e-means clustering with the l_1 and l_∞ norms. *IEEE Transactions on Systems, Man, and Cybernetics* **21**, pp. 545–554, 1991.

[Campello & Hruschka, 2006] Campello, R. J. G. B.; Hruschka, E. R. A fuzzy extension of the silhouette width criterion for cluster analysis. *Fuzzy Sets and Systems* **157**, pp. 2858–2875, 2006.

[Casillas et al., 2003] Casillas, A.; de Lena, M. T. G.; Martínez, R. Document clustering into an unknown number of clusters using a genetic algorithm. Em *TSD*, pp. 43–49, 2003.

[Cowgill et al., 1998] Cowgill, M. C.; Harvey, R. J.; Watson, L. T. A genetic algorithm approach to cluster analysis. Relatório técnico, Virginia Polytechnic Institute & State University, Blacksburg, VA, USA, 1998.

[Davies & Bouldin, 1979] Davies, D.; Bouldin, D. A cluster separation measure. *IEEE Transaction Patt. Anal. Mach. Intell.* **1**, pp. 224–227, 1979.

[Davis, 1996] Davis, L. *Handbook of Genetic Algorithms*. International Thomson Computer Press, 1996.

[Eiben & Smith, 2003] Eiben, A. E.; Smith, J. E. *Introduction to Evolutionary Computing*. Springer-Verlag Berlin, 2003.

[Estivill-Castro, 1997] Estivill-Castro, V. Spatial clustering for data mining with genetic algorithms. Relatório técnico, University of Queensland, , Australia, 1997.

[Everitt et al., 2001] Everitt, B. S.; Landau, S.; Leese, M. *Cluster Analysis*. Arnold Publishers, 2001.

[Falkenauer, 1998] Falkenauer, E. *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[Fogel, 1999] Fogel, D. B. *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. Wiley-IEEE Press, 1999.

[Fogel et al., 1966] Fogel, L.; Owens, A.; Walsh, M. *Artificial Intelligence through Simulated Evolution.* Wiley, 1966.

[Fränti et al., 1997] Fränti, P.; Kivijärvi, J.; Kaukoranta, T.; Nevalainen, O. Genetic algorithms for large-scale clustering problems. *Comput. J.* **40**(9), pp. 547–554, 1997.

[Handl & Knowles, 2005] Handl, J.; Knowles, J. Improvements to the scalability of multiobjective clustering. Em *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, pp. 2372–2379, Anaheim, CA, 2005.

[Holland, 1969] Holland, J. H. Adaptive plans optimal for payoff-only environments. Em *Proceedings of the 2nd Hawaii Int. Conf. on System Sciences*, 1969.

[Hruschka et al., 2006] Hruschka, E. R.; Campello, R. J.; de Castro, L. N. Evolving clusters in gene-expression data. *Information Sciences* **176**, pp. 1898–1927, 2006.

[Hruschka et al., 2004a] Hruschka, E. R.; Campello, R. J. G. B.; de Castro, L. N. Evolutionary search for optimal fuzzy c-means clustering. Em *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 685–690, 2004.

[Hruschka et al., 2004b] Hruschka, E. R.; Campello, R. J. G. B.; de Castro, L. N. Improving the efficiency of a clustering genetic algorithm. Em Lemaître, C.; Reyes, C. A.; González, J. A. (eds), *IBERAMIA*, volume 3315 of *Lecture Notes in Computer Science*, pp. 861–870. Springer, 2004.

[Hruschka et al., 2004c] Hruschka, E. R.; de Castro, L. N.; Campello, R. J. G. B. Evolutionary algorithms for clustering gene-expression data. Em *ICDM '04: Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pp. 403–406, Washington, DC, USA. IEEE Computer Society, 2004.

[Hruschka & Ebecken, 2003] Hruschka, E. R.; Ebecken, N. F. F. A genetic algorithm for cluster analysis. *Intell. Data Anal.* **7**(1), pp. 15–25, 2003.

[Jain & Dubes, 1988] Jain, A. K.; Dubes, R. C. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[Jain et al., 1999] Jain, A. K.; Murty, M. N.; Flynn, P. J. Data clustering: a review. *ACM Computing Surveys* **31**(3), pp. 264–323, 1999.

[Johnson & Wichern, 1998] Johnson, R. A.; Wichern, D. W. *Applied Multivariate Statistical Analysis*. Prentice-Hall,Inc., USA, 1998.

[Kaufman & Rousseeuw, 1990] Kaufman, L.; Rousseeuw, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley, 1990.

[Kivijärvi et al., 2003] Kivijärvi, J.; Fränti, P.; Nevalainen, O. Self-adaptive genetic algorithm for clustering. *Journal of Heuristics* **9**(2), pp. 113–129, 2003.

[Krishna & Murty, 1999] Krishna, K.; Murty, M. N. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* **29**(3), pp. 433–439, 1999.

[Liu et al., 2004] Liu, Y.; Chen, K.; Liao, X.; Zhang, W. A genetic clustering method for intrusion detection. *Pattern Recognition* **37**(5), pp. 927–942, 2004.

[Lopes, 2003] Lopes, S. *Bio*. Ed. Saraiva, Brasil, 2003.

[Lu et al., 2004a] Lu, Y.; Lu, S.; Fotouhi, F.; Deng, Y.; Brown, S. J. Fgka: a fast genetic k-means clustering algorithm. Em *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing*, pp. 622–623, New York, NY, USA. ACM Press, 2004.

[Lu et al., 2004b] Lu, Y.; Lu, S.; Fotouhi, F.; Deng, Y.; Brown, S. J. Incremental genetic k-means algorithm and its application in gene expression data analysis. *BMC Bioinformatics* **5**, pp. 172, 2004.

[Lucasius et al., 1993] Lucasius, C.; Dane, A.; Kateman, G. On k-medoid clustering of large data sets with the aid of a genetic algorithm: background, feasibility and comparison. *Analytica Chimica Acta* **282**, pp. 647–669, 1993.

[Maulik & Bandyopadhyay, 2000] Maulik, U.; Bandyopadhyay, S. Genetic algorithm-based clustering technique. *Pattern Recognition* **33**, pp. 1455–1465, 2000.

[Mitchell, 1996] Mitchell, M. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA, 1996.

[Naldi, 2006] Naldi, M. C., 2006. Agrupamento híbrido de dados utilizando algoritmos genéticos. Tese de mestrado, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.

[Pal & Bezdek, 1995] Pal, N. R.; Bezdek, J. C. On cluster validity for the fuzzy c-means model. *IEEE Transactions on Fuzzy Systems* **3**(3), pp. 370–379, 1995.

[Rand, 1971] Rand, W. M. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association* **66**, pp. 622–626, 1971.

[Rayward-Smith, 2005] Rayward-Smith, V. J. Metaheuristics for clustering in kdd. Em *Proceedings of the IEEE Congress on Evolutionary Computation 2005*, pp. 2380–2387. IEEE Press, 2005.

[Rechenberg, 1973] Rechenberg, I. *Evolutionsstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution.* Fromman-Hozlboog Verlag, 1973.

[Reinschmidt et al., 1999] Reinschmidt, J.; Gottschalk, H.; Kim, H.; Zwietering, D. *Intelligent Miner for Data: Enhance Your Business Intelligence*. Red Books IBM, California, USA, 1999.

[Schwefel, 1977] Schwefel, H. P. Numerische optimierung von computer-modellen mittels der evolutionsstrategie. *IRS Birkhaeuser* **26**, 1977.

[Sheng & Liu, 2004] Sheng, W.; Liu, X. A hybrid algorithm for k-medoid clustering of large data sets. Em *Proceedings of the Congress on Evolutionary Computation*, volume 1, pp. 77–82, 2004.

[Triola, 1999] Triola, M. F. *Introdução à estatística*. LTC, Brasil, 1999.

[Xu & Wunsch, 2005] Xu, R.; Wunsch, D. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on* **16**(3), pp. 645–678, 2005.

[Yeung et al., 2001] Yeung, K. Y.; Fraley, C.; Murua, A. E.; Raftery, A. E.; Ruzzo, W. L. Model-base clusterin and data transformations for gene expression data. *Bioinformatics* **17**(10), pp. 977–987, 2001.

[Yeung et al., 2003] Yeung, K. Y.; Medvedovic, M.; Bumgarner, R. E. Clustering gene-expression data with repeated measurements. *Genome Biology* **4**(5), 2003.

Apêndice A

Propriedades do *Rand Index* para grupos individuais

Nesse apêndice será demonstrado que o *Rand Index* de uma partição pode ser escrito como uma função dos *Rand Indexes* individuais de cada grupo.

O *Rand Index* é um critério numérico absoluto (externo) de validade de partições [Jain & Dubes, 1988]. Para formalizá-lo considere-se um conjunto de dados **X** formado por N objetos, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, para o qual se conhece *a priori* o conjunto de k classes $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ que descrevem a estrutura do conjunto **X**, sendo que $\{\mathbf{u}_1 \cup, \ldots, \cup \mathbf{u}_k\} = \mathbf{X}$ e $\mathbf{u}_i \cap \mathbf{u}_j = \emptyset$ para $\forall i, j \in \{1, \ldots, k\}$ e $i \neq j$. Pode-se avaliar uma partição formada por k' grupos disjuntos $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_k\text{'}\}$ em relação à partição de referência **U** por meio do *Rand Index*. A partição de referência, **U**, pode ser vista como uma partição ideal a ser obtida por meio de um método de agrupamento de dados. Assume-se que este obteve a partição **V**. Deseja-se, portanto, medir a compatibilidade (aderência) entre estas partições. Formalmente o *Rand Index* [Rand, 1971] pode ser escrito como:

$$\Omega(\mathbf{U}, \mathbf{V}) = \frac{a + d}{a + b + c + d} \tag{A.1}$$

onde:

- a: Metade do número de pares ordenados de objetos pertencentes ao mesmo grupo em **V** e à mesma classe em **U**;
- b: Metade do número de pares ordenados de objetos pertencentes a grupos distintos em **V** e à mesma classe em **U**;
- c: Metade do número de pares ordenados de objetos pertencentes ao mesmo grupo em **V** e a

classes distintas em **U**;

- d: Metade do número de pares ordenados de objetos pertencentes a grupos distintos em **V** e a classes distintas em **U**.

Os termos a e d da equação (A.1) correspondem às quantidades de pares de objetos agrupados corretamente na partição **V** sob a ótica de classificação (partição de referência **U**), enquanto os termos b e c representam a quantidade de pares de objetos agrupados erroneamente. Ω é uma função monotônica em relação à compatibilidade dos conjuntos **U** e **V** de tal forma que $\Omega \in [0, 1]$. Particularmente quando $b = c = 0$, então $\Omega = 1$ e portanto, as partições **U** e **V** são idênticas. Quando $a = d = 0$, então $\Omega = 0$ e portanto, **U** e **V** são completamente incompatíveis (não apresentam nenhuma aderência) segundo o *Rand Index* .

Normalmente, o *Rand Index* é calculado sobre $\frac{N^2-N}{2}$ pares *não ordenados* de objetos, resultantes de uma combinação simples dos N objetos do conjunto em pares. No entanto, nesta demonstração considera-se que os pares são formados pela combinação de todos os objetos em um arranjo simples, resultando em $N^2 - N$ pares *ordenados* de objetos. Em decorrência desta imposição, os pares de objetos são contados em metade da sua quantidade afim de manter a compatibilidade com a definição original do *Rand Index* . A opção por calcular o *Rand Index* sobre os $N^2 - N$ pares ordenados de objetos é motivada pela simplificação dos cálculos contidos nessa demonstração.

O *Rand Index* pode ser escrito em função das contribuições parciais de cada grupo para o índice geral da partição. Para tal, define-se o *Rand Index* para um determinado grupo $\mathbf{v}_i \in \mathbf{V}$ como:

$$\Omega_i(\mathbf{U}, \mathbf{V}) = \frac{a_i + d_i}{a_i + b_i + c_i + d_i} \tag{A.2}$$

onde:

- a_i: Metade do número de pares de objetos pertencentes ao grupo $\mathbf{v}_i$ e à mesma classe em **U**;

- b_i: Metade do número de pares de objetos que pertencem a grupos distintos $\mathbf{v}_i$ e $\mathbf{v}_j$, ($\forall j \in \{1, \ldots, k'\}$, $i \neq j$, $\mathbf{v}_j \in \mathbf{V}$),e à mesma classe em **U**;

- c_i: Metade do número de pares de objetos que pertencem ao grupo $\mathbf{v}_i$ e a classes distintas em **U**;

- d_i: Metade do número de pares de objetos que pertencem a grupos distintos $\mathbf{v}_i$ e $\mathbf{v}_j$,($\forall j \in \{1, \ldots, k'\}$, $i \neq j$, $\mathbf{v}_j \in \mathbf{V}$), e a classes distintas em **U**.

Será demonstrado a seguir que o *Rand Index* total de uma partição é a média ponderada dos *Rand Indexes* de cada um dos grupos $\mathbf{v}_i$, $i \in \{1, \ldots, k'\}$. Ou seja:

Proposição1:

$$\Omega(\mathbf{U},\mathbf{V}) = \frac{\sum_{i=1}^{k'} \Omega_i(\mathbf{U},\mathbf{V})\omega_i}{\sum_{i=1}^{k'} \omega_i},$$

onde

$$\Omega_i(\mathbf{U},\mathbf{V}) = \frac{a_i + d_i}{a_i + b_i + c_i + d_i},\ \omega_i = (a_i + b_i + c_i + d_i)$$

Para simplificar a demonstração, será adotada a seguinte notação:

- $C(\mathbf{x}_i)$ representa o rótulo da classe em **U** ao qual pertence o objeto $\mathbf{x}_i$ do conjunto **X**;
- $G(\mathbf{x}_i)$ representa o rótulo do grupo em **V** ao qual pertence o objeto $\mathbf{x}_i$ do conjunto **X**;
- O conjunto **X** é ordenado de tal forma que os objetos pertencentes a um grupo $\mathbf{v}_i$ são apresentados da posição p_i da lista ordenada até a posição N_i. Assume-se que $p_1 = 1$, assim $p_i = (N_{i-1} + 1)$, para $i > 1$ e $N_i = (p_i + |\mathbf{v}_i| - 1)$.

Inicialmente será demonstrado que as somatórias dos termos a_i, b_i, c_i e d_i para $i = \{1, \ldots, k'\}$ são equivalentes aos termos a, b, c, d da partição total respectivamente. Ou seja $a = \sum_{m=1}^{k'} a_m$, $b = \sum_{m=1}^{k'} b_m$, $c = \sum_{m=1}^{k'} c_m$ e $d = \sum_{m=1}^{k'} d_m$.

Para demonstrar que $a = \sum_{m=1}^{k'} a_m$ considere-se que o termo a da equação (A.1) pode ser definido como:

$$a = \sum_{i=1}^{N} \sum_{\substack{j=1,\\ j\neq i}}^{N} f_1(\mathbf{x}_i, \mathbf{x}_j) \tag{A.3}$$

$$f_1(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \frac{1}{2}, \text{ se } G(\mathbf{x}_i) = G(\mathbf{x}_j) \wedge C(\mathbf{x}_i) = C(\mathbf{x}_j) \\ 0, \text{ caso contrário.} \end{cases} \tag{A.4}$$

O termo a_i da equação (A.2), por sua vez, pode ser definido como:

$$a_i = \sum_{j=p_i}^{N_i} \sum_{\substack{l=p_i,\\ l\neq j}}^{N_i} f_1(\mathbf{x}_j, \mathbf{x}_l) \tag{A.5}$$

É possível reescrever o termo a definido na equação (A.3) em função dos grupos da partição **V**:

$$\begin{aligned} a = &\sum_{i_1=p_1}^{N_1}\sum_{\substack{j_1=p_1,\\ j_1\neq i_1}}^{N_1} f_1(\mathbf{x}_{i_1},\mathbf{x}_{j_1}) + \sum_{i_1=p_1}^{N_1}\sum_{j_2=p_2}^{N_2} f_1(\mathbf{x}_{i_1},\mathbf{x}_{j_2}) + \ldots + \sum_{i_1=p_1}^{N_1}\sum_{j_{k'}=p_{k'}}^{N_{k'}} f_1(\mathbf{x}_{i_1},\mathbf{x}_{j_{k'}}) + \\ &+ \sum_{i_2=p_2}^{N_2}\sum_{j_1=p_1}^{N_1} f_1(\mathbf{x}_{i_2},\mathbf{x}_{j_1}) + \sum_{i_2=p_2}^{N_2}\sum_{\substack{j_2=p_2,\\ j_2\neq i_2}}^{N_2} f_1(\mathbf{x}_{i_2},\mathbf{x}_{j_2}) + \ldots + \sum_{i_2=p_2}^{N_2}\sum_{j_{k'}=p_{k'}}^{N_{k'}} f_1(\mathbf{x}_{i_2},\mathbf{x}_{j_{k'}}) + \ldots + \\ &+ \sum_{i_{k'}=p_{k'}}^{N_{k'}}\sum_{j_1=p_1}^{N_1} f_1(\mathbf{x}_{i_{k'}},\mathbf{x}_{j_1}) + \sum_{i_{k'}=p_{k'}}^{N_{k'}}\sum_{j_2=p_2}^{N_2} f_1(\mathbf{x}_{i_{k'}},\mathbf{x}_{j_2}) + \ldots + \sum_{i_{k'}=p_{k'}}^{N_{k'}}\sum_{\substack{j_{k'}=p_{k'},\\ j_{k'}\neq i_{k'}}}^{N_{k'}} f_1(\mathbf{x}_{i_{k'}},\mathbf{x}_{j_{k'}}) \end{aligned} \tag{A.6}$$

Na equação (A.6), note que os índices adotados foram padronizados de tal forma que o primeiro objeto do par ordenado na função f_1 é indexado por i_x e o segundo objeto do par ordenado é indexado por j_y, sendo nesse caso x e y referentes aos grupos $\mathbf{v}_x$ e $\mathbf{v}_y$ aos quais pertencem tais objetos. Tal padronização foi utilizada unicamente com o intuito de facilitar o entendimento das expressões e equações aqui apresentadas.

Ainda na equação (A.6), os somatórios que denotam o cálculo de a para objetos que pertencem a diferentes grupos serão necessariamente nulos em razão da equação (A.4). Ignorando os somatórios nulos, o termo a pode ser escrito como:

$$a = \sum_{i_1=p_1}^{N_1}\sum_{\substack{j_1=p_1,\\ j_1\neq i_1}}^{N_1} f_1(\mathbf{x}_{i_1},\mathbf{x}_{j_1}) + \sum_{i_2=p_2}^{N_2}\sum_{\substack{j_2=p_2,\\ j_2\neq i_2}}^{N_2} f_1(\mathbf{x}_{i_2},\mathbf{x}_{j_2}) + \ldots + \sum_{i_{k'}=p_{k'}}^{N_{k'}}\sum_{\substack{j_{k'}=p_{k'},\\ j_{k'}\neq i_{k'}}}^{N_{k'}} f_1(\mathbf{x}_{i_{k'}},\mathbf{x}_{j_{k'}}) \tag{A.7}$$

Pode-se na equação (A.7) substituir os somatórios com $\sum_{i_m=p_m}^{N_m}\sum_{\substack{j_m=p_m,\\ j_m\neq i_m}}^{N_m} f_1(\mathbf{x}_{i_m},\mathbf{x}_{j_m})$ por a_m levando a:

$$a = a_1 + a_2 + \ldots + a_{k'} = \sum_{m=1}^{k'} a_m \tag{A.8}$$

C.Q.D

De maneira análoga, é possível demonstrar que $b = \sum_{m=1}^{k'} b_m$. Para tal, considere-se que b pode ser definido a partir da equação (A.1) como:

$$b = \sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N} f_2(\mathbf{x}_i,\mathbf{x}_j) \tag{A.9}$$

$$f_2(\mathbf{x}_i,\mathbf{x}_j) = \begin{cases} \frac{1}{2}, \text{ se } G(\mathbf{x}_i)\neq G(\mathbf{x}_j) \wedge C(\mathbf{x}_i) = C(\mathbf{x}_j) \\ 0, \text{ caso contrário.} \end{cases} \tag{A.10}$$

O termo b_i da equação (A.2), por sua vez, pode ser definido como:

$$b_i = \sum_{j=p_i}^{N_i} \sum_{\substack{q=1,\\ q\neq i}}^{k'} \sum_{l=p_q}^{N_q} f_2(\mathbf{x}_j, \mathbf{x}_l) \tag{A.11}$$

É possível reescrever o termo b definido na equação (A.9) em função dos grupos da partição **V**:

$$\begin{aligned} b = &\sum_{i_1=p_1}^{N_1} \sum_{\substack{j_1=p_1,\\ j_1\neq i_1}}^{N_1} f_2(\mathbf{x}_{i_1}, \mathbf{x}_{j_1}) + \sum_{i_1=p_1}^{N_1} \sum_{j_2=p_2}^{N_2} f_2(\mathbf{x}_{i_1}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_1=p_1}^{N_1} \sum_{j_{k'}=p_{k'}}^{N_{k'}} f_2(\mathbf{x}_{i_1}, \mathbf{x}_{j_{k'}}) + \\ &+ \sum_{i_2=p_2}^{N_2} \sum_{j_1=p_1}^{N_1} f_2(\mathbf{x}_{i_2}, \mathbf{x}_{j_1}) + \sum_{i_2=p_2}^{N_2} \sum_{\substack{j_2=p_2,\\ j_2\neq i_2}}^{N_2} f_2(\mathbf{x}_{i_2}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_2=p_2}^{N_2} \sum_{j_{k'}=p_{k'}}^{N_{k'}} f_2(\mathbf{x}_{i_2}, \mathbf{x}_{j_{k'}}) + \ldots + \\ &+ \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_1=p_1}^{N_1} f_2(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_1}) + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_2=p_2}^{N_2} f_2(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{\substack{j_{k'}=p_{k'},\\ j_{k'}\neq i_{k'}}}^{N_{k'}} f_2(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_{k'}}) \end{aligned} \tag{A.12}$$

Na equação (A.12), os somatórios que denotam o cálculo de b para objetos que pertencem aos mesmos grupos serão necessariamente nulos em razão da equação (A.10). Ignorando os somatórios nulos, o termo b pode ser escrito como:

$$\begin{aligned} b = &\sum_{i_1=p_1}^{N_1} \sum_{j_2=p_2}^{N_2} f_2(\mathbf{x}_{i_1}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_1=p_1}^{N_1} \sum_{j_{k'}=p_{k'}}^{N_{k'}} f_2(\mathbf{x}_{i_1}, \mathbf{x}_{j_{k'}}) + \\ &+ \sum_{i_2=p_2}^{N_2} \sum_{j_1=p_1}^{N_1} f_2(\mathbf{x}_{i_2}, \mathbf{x}_{j_1}) + \sum_{i_2=p_2}^{N_2} \sum_{j_3=p_3}^{N_3} f_2(\mathbf{x}_{i_2}, \mathbf{x}_{j_3}) + \ldots + \sum_{i_2=p_2}^{N_2} \sum_{j_{k'}=p_{k'}}^{N_{k'}} f_2(\mathbf{x}_{i_2}, \mathbf{x}_{j_{k'}}) + \ldots + \\ &+ \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_1=p_1}^{N_1} f_2(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_1}) + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_2=p_2}^{N_2} f_2(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_{k'-1}=p_{k'-1}}^{N_{k'-1}} f_2(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_{k'-1}}) \end{aligned} \tag{A.13}$$

Pode-se na equação (A.13) substituir os somatórios com $\sum_{i_m=p_m}^{N_m} \sum_{j_o=p_o}^{N_0} f_2(\mathbf{x}_{i_m}, \mathbf{x}_{j_o}) + \ldots +$ $+ \sum_{i_m=p_m}^{N_m} \sum_{j_{m-1}=p_{m-1}}^{N_{m-1}} f_2(\mathbf{x}_{i_m}, \mathbf{x}_{j_{m-1}}) + \sum_{i_m=p_m}^{N_m} \sum_{j_{m+1}=p_{m+1}}^{N_{m+1}} f_2(\mathbf{x}_{i_m}, \mathbf{x}_{j_{m+1}}) + \ldots + \sum_{i_m=p_m}^{N_m} \sum_{j_q=p_q}^{N_q} f_2(\mathbf{x}_{i_m}, \mathbf{x}_{j_q})$, onde:

$$o = min\left\{1, \ldots, m-1, m+1, \ldots, k'\right\}$$

e

$$q = max\left\{1, \ldots, m-1, m+1, \ldots, k'\right\}$$

por b_m levando a:

$$b = b_1 + b_2 + \ldots + b_{k'} = \sum_{m=1}^{k'} b_m \tag{A.14}$$

C.Q.D

Demonstrações análogas àquelas realizadas para $a = \sum_{m=1}^{k'} a_m$ e $b = \sum_{m=1}^{k'} b_m$ podem ser obtidas para $c = \sum_{m=1}^{k'} c_m$ e $d = \sum_{m=1}^{k'} d_m$ definidos na equação (A.1).

Em particular, para a demonstração de $c = \sum_{m=1}^{k'} c_m$, considere-se que o termo c da equação (A.1) pode ser definido como:

$$c = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} f_3(\mathbf{x}_i, \mathbf{x}_j) \tag{A.15}$$

$$f_3(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \frac{1}{2}, \text{ se } G(\mathbf{x}_i) = G(\mathbf{x}_j) \wedge C(\mathbf{x}_i) \neq C(\mathbf{x}_j) \\ 0, \text{ caso contrário.} \end{cases} \tag{A.16}$$

O termo c_i da equação (A.2), por sua vez, pode ser definido como:

$$c_i = \sum_{j=p_i}^{N_i} \sum_{\substack{l=p_i \\ l \neq j}}^{N_i} f_3(\mathbf{x}_j, \mathbf{x}_l) \tag{A.17}$$

É possível reescrever o termo a definido na equação (A.15) em função dos grupos da partição **V**:

$$\begin{aligned} c = & \sum_{i_1=p_1}^{N_1} \sum_{\substack{j_1=p_1, \\ j_1 \neq i_1}}^{N_1} f_3(\mathbf{x}_{i_1}, \mathbf{x}_{j_1}) + \sum_{i_1=p_1}^{N_1} \sum_{j_2=p_2}^{N_2} f_3(\mathbf{x}_{i_1}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_1=p_1}^{N_1} \sum_{j_{k'}=p_{k'}}^{N_{k'}} f_3(\mathbf{x}_{i_1}, \mathbf{x}_{j_{k'}}) + \\ & + \sum_{i_2=p_2}^{N_2} \sum_{j_1=p_1}^{N_1} f_3(\mathbf{x}_{i_2}, \mathbf{x}_{j_1}) + \sum_{i_2=p_2}^{N_2} \sum_{\substack{j_2=p_2, \\ j_2 \neq i_2}}^{N_2} f_3(\mathbf{x}_{i_2}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_2=p_2}^{N_2} \sum_{j_{k'}=p_{k'}}^{N_{k'}} f_3(\mathbf{x}_{i_2}, \mathbf{x}_{j_{k'}}) + \ldots + \\ & + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_1=p_1}^{N_1} f_3(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_1}) + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_2=p_2}^{N_2} f_3(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{\substack{j_{k'}=p_{k'}, \\ j_{k'} \neq i_{k'}}}^{N_{k'}} f_3(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_{k'}}) \end{aligned} \tag{A.18}$$

Na equação (A.18), os somatórios que denotam o cálculo de c para objetos que pertencem a diferentes grupos serão necessariamente nulos em razão da equação (A.16). Ignorando os somatórios nulos, o termo c pode ser escrito como:

$$c = \sum_{i_1=p_1}^{N_1} \sum_{\substack{j_1=p_1, \\ j_1 \neq i_1}}^{N_1} f_3(\mathbf{x}_{i_1}, \mathbf{x}_{j_1}) + \sum_{i_2=p_2}^{N_2} \sum_{\substack{j_2=p_2, \\ j_2 \neq i_2}}^{N_2} f_3(\mathbf{x}_{i_2}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{\substack{j_{k'}=p_{k'}, \\ j_{k'} \neq i_{k'}}}^{N_{k'}} f_3(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_{k'}}) \tag{A.19}$$

Pode-se na equação (A.19) substituir os somatórios com $\sum_{i_m=p_m}^{N_m} \sum_{\substack{j_m=p_m, \\ j_m \neq i_m}}^{N_m} f_3(\mathbf{x}_{i_m}, \mathbf{x}_{j_m})$ por c_m levando a:

$$c = c_1 + c_2 + \ldots + c_{k'} = \sum_{m=1}^{k'} c_m \tag{A.20}$$

C.Q.D

De maneira análoga, é possível demonstrar que $d = \sum_{m=1}^{k'} d_m$. Para tal, considere-se que d pode ser definido a partir da equação (A.1) como:

$$d = \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} f_4(\mathbf{x}_i, \mathbf{x}_j) \tag{A.21}$$

$$f_4(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \frac{1}{2}, \text{ se } G(\mathbf{x}_i) \neq G(\mathbf{x}_j) \wedge C(\mathbf{x}_i) \neq C(\mathbf{x}_j) \\ 0, \text{ caso contrário.} \end{cases} \tag{A.22}$$

O termo d_i da equação (A.2), por sua vez, pode ser definido como:

$$d_i = \sum_{j=p_i}^{N_i} \sum_{\substack{q=1, \\ q \neq i}}^{k'} \sum_{l=p_q}^{N_q} f_4(\mathbf{x}_j, \mathbf{x}_l) \tag{A.23}$$

É possível reescrever o termo d definido na equação (A.21) em função dos grupos da partição **V**:

$$\begin{aligned} d = & \sum_{i_1=p_1}^{N_1} \sum_{\substack{j_1=p_1, \\ j_1 \neq i_1}}^{N_1} f_4(\mathbf{x}_{i_1}, \mathbf{x}_{j_1}) + \sum_{i_1=p_1}^{N_1} \sum_{j_2=p_2}^{N_2} f_4(\mathbf{x}_{i_1}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_1=p_1}^{N_1} \sum_{j_{k'}=p_{k'}}^{N_{k'}} f_4(\mathbf{x}_{i_1}, \mathbf{x}_{j_{k'}}) + \\ & + \sum_{i_2=p_2}^{N_2} \sum_{j_1=p_1}^{N_1} f_4(\mathbf{x}_{i_2}, \mathbf{x}_{j_1}) + \sum_{i_2=p_2}^{N_2} \sum_{\substack{j_2=p_2, \\ j_2 \neq i_2}}^{N_2} f_4(\mathbf{x}_{i_2}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_2=p_2}^{N_2} \sum_{j_{k'}=p_{k'}}^{N_{k'}} f_4(\mathbf{x}_{i_2}, \mathbf{x}_{j_{k'}}) + \ldots + \\ & + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_1=p_1}^{N_1} f_4(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_1}) + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{j_2=p_2}^{N_2} f_4(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_2}) + \ldots + \sum_{i_{k'}=p_{k'}}^{N_{k'}} \sum_{\substack{j_{k'}=p_{k'}, \\ j_{k'} \neq i_{k'}}}^{N_{k'}} f_4(\mathbf{x}_{i_{k'}}, \mathbf{x}_{j_{k'}}) \end{aligned} \tag{A.24}$$

Na equação (A.24), os somatórios que denotam o cálculo de d para objetos que pertencem aos mesmos grupos serão necessariamente nulos em razão da equação (A.22). Ignorando os somatórios

nulos, o termo d pode ser escrito como:

$$d = \sum_{i_1=p_1}^{N_1}\sum_{j_2=p_2}^{N_2} f_4(\mathbf{x}_{i_1},\mathbf{x}_{j_2}) + \ldots + \sum_{i_1=p_1}^{N_1}\sum_{j_{k'}=p_{k'}}^{N_{k'}} f_4(\mathbf{x}_{i_1},\mathbf{x}_{j_{k'}}) +$$
$$+ \sum_{i_2=p_2}^{N_2}\sum_{j_1=p_1}^{N_1} f_4(\mathbf{x}_{i_2},\mathbf{x}_{j_1}) + \sum_{i_2=p_2}^{N_2}\sum_{j_3=p_3}^{N_3} f_4(\mathbf{x}_{i_2},\mathbf{x}_{j_3}) + \ldots + \sum_{i_2=p_2}^{N_2}\sum_{j_{k'}=p_{k'}}^{N_{k'}} f_4(\mathbf{x}_{i_2},\mathbf{x}_{j_{k'}}) + \ldots +$$
$$+ \sum_{i_{k'}=p_{k'}}^{N_{k'}}\sum_{j_1=p_1}^{N_1} f_4(\mathbf{x}_{i_{k'}},\mathbf{x}_{j_1}) + \sum_{i_{k'}=p_{k'}}^{N_{k'}}\sum_{j_2=p_2}^{N_2} f_4(\mathbf{x}_{i_{k'}},\mathbf{x}_{j_2}) + \ldots + \sum_{i_{k'}=p_{k'}}^{N_{k'}}\sum_{j_{k'-1}=p_{k'-1}}^{N_{k'-1}} f_4(\mathbf{x}_{i_{k'}},\mathbf{x}_{j_{k'-1}}) \tag{A.25}$$

Pode-se na equação (A.25) substituir os somatórios com $\sum_{i_m=p_m}^{N_m}\sum_{j_o=p_o}^{N_0} f_4(\mathbf{x}_{i_m},\mathbf{x}_{j_o}) + \ldots +$ $+ \sum_{i_m=p_m}^{N_m}\sum_{j_{m-1}=p_{m-1}}^{N_{m-1}} f_4(\mathbf{x}_{i_m},\mathbf{x}_{j_{m-1}}) + \sum_{i_m=p_m}^{N_m}\sum_{j_{m+1}=p_{m+1}}^{N_{m+1}} f_4(\mathbf{x}_{i_m},\mathbf{x}_{j_{m+1}}) + \ldots + \sum_{i_m=p_m}^{N_m}\sum_{j_q=p_q}^{N_q} f_4(\mathbf{x}_{i_m},\mathbf{x}_{j_q})$, onde:

$$o = min\left\{1,\ldots,m-1,m+1,\ldots,k'\right\}$$

e

$$q = max\left\{1,\ldots,m-1,m+1,\ldots,k'\right\}$$

por d_m levando a:

$$d = d_1 + d_2 + \ldots + d_{k'} = \sum_{m=1}^{k'} d_m \tag{A.26}$$

C.Q.D

A partir da demonstração de que $a = \sum_{m=1}^{k'} a_m$, $b = \sum_{m=1}^{k'} b_m$, $c = \sum_{m=1}^{k'} c_m$ e $d = \sum_{m=1}^{k'} d_m$ e da definição de Ω na equação (A.1) temos que:

$$\Omega(\mathbf{U},\mathbf{V}) = \frac{\sum_{m=1}^{k'} a_m + \sum_{m=1}^{k'} d_m}{\sum_{m=1}^{k'} a_m + \sum_{m=1}^{k'} b_m + \sum_{m=1}^{k'} c_m + \sum_{m=1}^{k'} d_m} = \frac{\sum_{m=1}^{k'} (a_m + d_m)}{\sum_{m=1}^{k'} (a_m + b_m + c_m + d_m)}$$

Lembrando que na **Proposição1** foi definido que $\omega_i = (a_i + b_i + c_i + d_i)$, pode-se descrever o *Rand Index* como:

$$\Omega(\mathbf{U},\mathbf{V}) = \frac{\sum_{m=1}^{k'} (a_m + d_m)}{\sum_{m=1}^{k'} \omega_i}$$

Reescrevendo o *Rand Index* para um grupo em função de ω_i temos:

$$\Omega_i(\mathbf{U}, \mathbf{V}) = (a_i + d_i)\omega_i^{-1}$$

Sendo $a_i + d_i = \Omega_i(\mathbf{U}, \mathbf{V})\omega_i$, pode-se definir o *Rand Index* em função dos índices parciais como:

$$\Omega(\mathbf{U}, \mathbf{V}) = \frac{\sum_{m=1}^{k'} \Omega_m(\mathbf{U}, \mathbf{V})\omega_m}{\sum_{m=1}^{k'} \omega_m} \tag{A.27}$$

C.Q.D